Universidad de Guanajuato / División de Ingenierías

Aplicación de «Redes Neuronales Convolucionales» en la identificación y clasificación de barreras metálicas en carreteras de cuerpo único.

Comité de tesis

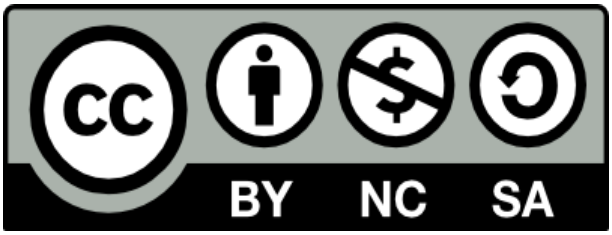
Tesista: Rangel Rivera Ulises Osmar.

Director: Mtro. Jacob Esaú Salazar Solano.

Codirector: Mtro. Alfonso Ceseña Quiñones.

Codirector: Dr. Salvador Botello Rionda.

14 de octubre de 2024


BY NC SA

Índice general

Índice de figuras

Reconocimientos

Quiero agradecer a mis asesores quienes me brindaron su apoyo intelectual en la realización de este trabajo.
Quiero agradecer su compañía a través de estos años de aciertos y errores, los cuales me han no solo ayudado a culminar este proyecto, pero a mejorar como profesional y persona.
Al Dr. Salvador Botello Rionda por su apoyo al informarme de herramientas que podría utilizar para desarrollar el sistema, su sinceridad y sus buenos consejos durante la realización de esta investigación.
Al Mtro. Alfonso Ceseña Quiñones por sus excelentes sugerencias de fuentes de información con las cuales pude expandir mi entendimiento del tema.
Al Mtro. Jacob Esaú Salazar Solano por su profundo involucramiento en cada uno de los detalles de este proyecto, sin su guía este proyecto no habría podido realizarse.
A Gloria Almanza García pues gracias a ella tuve la oportunidad de aprender sobre el sistema empleado en esta investigación.
A Ilse Viridiana Huerta Ramírez, por su excelente e invaluable apoyo en la revisión de este texto.
Al Laboratorio de Supercomputo del Bajío. Sin su apoyo este trabajo no habría sido posible.
El agradecimiento especial va dirigido a mi familia por su paciencia y comprensión. Sin su apoyo moral me habría dado por vencido hace muchos años. Me han motivado a superarme en todo aspecto, a perseguir los temas que realmente me apasionan y a no darme por vencido cuando parece no haber salida.

Resumen

El avance de la tecnología nos brinda herramientas que ofrecen nuevas alternativas para resolver problemas, esto a veces tiene el potencial de cambiar la manera de llevarlas acabo como se presenta en esta investigación.
Actualmente en el país la Ingeniería Civil aún ejecuta muchas actividades con métodos manuales; ya sea medición, evaluación o ejecución las cuales se pueden eficientar con tecnología.
Con base en lo anterior, se tomó la decisión de llevar a cabo el presente trabajo de investigación y desarrollo con el fin de eficientar los procesos del inventariado de dispositivos de seguridad en la red carretera estatal (en específico una de las actividades consiste en trabajar con las carreteras secundarias del estado de Guanajuato). La actividad descrita es realizada por la Secretaría de Comunicaciones y Transportes (SCT).
La herramienta empleada fue una Red Neuronal Convolucional (RNC) o, por su nombre en inglés, *Convolutional Neural Network (CNN)*. Esta herramienta no pertenece al arsenal habitual del ingeniero civil, pues viene del área de la inteligencia artificial, rama del conocimiento que ha tenido un desarrollo explosivo en estos últimos años.
Se eligió este método por su desempeño excepcional en la clasificación de imágenes, al extraer rasgos que facilitan la detección automática de los elementos que le son proporcionados a la red. En este caso, imágenes de las barreras metálicas a las orillas de la corona de las carreteras secundarias de cuerpo único.
La implementación se basa en un software para el llenado automático del formato de Excel que se utiliza en la SCT. El software se desarrolló en el lenguaje de programación *Python*.
En resumen, los resultados obtenidos al inventariar cuatro carreteras secundarias reales fueron una reducción del 90.21 % en el tiempo de ejecución, esto incluye la detección de barreras (o defensas) metálicas y su registro en un formato de Excel. También se logró una exhaustividad del 93 % con la red seleccionada, es decir que de las 232 imágenes de barreras que existían en el conjunto a clasificar de 10,976 imágenes (cantidad de fotos en las cuatro bases de datos usadas), se lograron registrar y clasificar correctamente 216 (incluyendo imágenes con un nivel de obstrucción alto y visibilidad pobre).

Palabras clave: Carreteras, Dispositivos de seguridad, Aprendizaje de Máquina, Redes Neuronales Artificiales Convolucionales, Redes Neuronales Artificiales.

Abstract

Currently, many civil engineering activities in Mexico are still being performed in a traditional, manual manner, without the use of software tools to reduce the time required to do them.

Taking this information into account, this research and development project aims to optimize the process of completing the inventory of road safety devices, which is performed by the Secretariat of Infrastructure, Communications, and Transportation in Mexico using road images from the state of Guanajuato, with a focus solely on secondary roads.

The tool selected for this task was an Artificial Convolutional Neural Network (abbreviated as ACNN or CNN). The choice of this method was based on its exceptional performance in image classification, as it learns features that enable it to identify objects of interest in real-world images presented to the network. This approach is uncommon in the toolkit of a civil engineer, as it originates from the field of artificial intelligence, whose tools have been applied to a wide range of disciplines in recent years.

The Convolutional Neural Network architecture designed in this research paper is accompanied by a python script that creates the Excel inventory file, fills the blanks, gives the file a format similar to the one used in the secretariat and returns the partially filled inventory to the user.

After processing four real secondary roads using the tools developed in this research, we observed a 90.21 % reduction in execution time, including the classification of metal crash barriers and their registration in the Excel format. The selected CNN also achieved a recall of 93 %, meaning that out of 232 metal crash barriers present in the set of 10,976 road images, our system correctly registered and classified 216. Some of these images had significant obstructions and poor visibility.

Keywords: Roads, Road Safety Devices, Machine Learning, Artificial Convolutional Neural Networks, Artificial Neural Networks.

Capítulo 1

Planteamiento del problema

El problema por resolver surge de una tarea rutinaria y repetitiva realizada por la SCT (Secretaría de Comunicaciones y Transportes) con el propósito de dar mantenimiento a las carreteras del país. La actividad en cuestión se llama "Actualización del inventariado de dispositivos de seguridad y detección de elementos dañados o que no cumplen con la normatividad en la red carretera estatal ". Esta consiste en realizar un inventario en donde se registran: todas las barreras metálicas de orilla de corona, las barreras de concreto, las terminales de las barreras, las secciones de transición en las barreras, entre otros elementos variados que forman parte de las estructuras de seguridad vial en las carreteras.
La actividad se realiza a nivel nacional, con cada estado participando en el inventariado de las estructuras que se encuentran en los límites geográficos de la red nacional que le corresponde. Como consecuencia, tenemos a muchos ingenieros y/o practicantes trabajando en esta actividad, que se puede extender por semanas o meses debido a las correcciones. Este procedimiento consiste en:

1. Identificar visualmente en imágenes todos los elementos mencionados (a través de un programa que nos permite realizar un recorrido de la carretera, del cual podemos ver un ejemplo en el anexo E).

2. Vaciar esta información manualmente en una base de datos de Excel que podemos ver en el anexo F.

Este proceso es lento. En el estado de Guanajuato se programa para 5 meses, más 2 meses de revisión aproximadamente, donde se verifica que la información cumpla con las especificaciones del Manual de Señalización Vial y Dispositivos de Seguridad. [1]

En esta investigación nos enfocamos en demostrar que se puede agilizar el proceso de registro e identificación de barreras metálicas en carreteras secundarias. Se utilizan estas carreteras pues es donde se pueden cometer más errores a la hora de detectar barreras, al estar cubiertas de maleza u otras obstrucciones que no se presentan con la misma frecuencia en otras clases de carreteras (como básicas o autopistas).

[1] Estas aproximaciones se obtuvieron a través de preguntas dirigidas a un ingeniero dentro de la SCT encargado de la actividad.

Capítulo 2

Justificación

La necesidad de realizar este proyecto es permitirle al ingeniero civil emplear sus habilidades y conocimientos en tareas que realmente los requieran, lo cual agilizaría, al menos en este caso, los procedimientos de la Secretaría de Comunicaciones y Transportes. Esto nos beneficia a todos, pues dependemos de sus servicios para la creación de infraestructura, su mantenimiento y evaluación, para garantizar las condiciones que nos permiten transportarnos con seguridad como usuarios de la red carretera bajo su jurisdicción.

Las ramificaciones de la mejora en los procesos de una dependencia nacional van más allá de un beneficio local a sus usuarios, pues repercuten a gran escala en la economía del país.

Otra razón para realizar este trabajo es establecer una conexión entre la Ingeniería Civil y otras áreas del conocimiento como la programación y la inteligencia artificial, para demostrar la existencia de métodos computacionales aplicables a tareas de ingeniería que actualmente se ejecutan con métodos rudimentarios.

Por último, el desarrollo de este proyecto abre las puertas para que otros estudiantes puedan expandir y explorar las ideas presentadas. Esto les brindará una nueva forma de resolver problemas de ingeniería, más allá de la Ingeniería Civil.

Capítulo 3

Hipótesis

Es posible automatizar el inventariado de barreras metálicas utilizando una Red Neuronal Convolucional (RNC).

Capítulo 4

Antecedentes

4.1 Antecedentes del tema

En esta sección se mencionan algunos trabajos relacionados con la identificación de elementos de seguridad en carreteras. Tres de ellos toman en cuenta el método de aprendizaje *transfer learning*, el cual, según Wikipedia contributors (2022), se enfoca en almacenar el conocimiento adquirido resolviendo un problema y aplicarlo en otro diferente. Uno de los trabajos hace una comparación entre este método y el de *non-transfer learning* (aprendizaje sin transferencia). Este método no utiliza como base a otro modelo, como en el caso del *non-transfer learning*, sino que solo se usa lo aprendido durante el entrenamiento para resolver el problema.

Comencemos por la investigación de Sainju, A. M. and Jiang, Z. (2020) donde proponen un sistema que implementa CNN (*Convolutional Neural Networks*) y LSTM (*Long Short-Term Memory*) para realizar un mapa de las características de seguridad vial empleando imágenes del sistema de *Google Street View*. En el artículo usan los indicadores de desempeño *F-score* (valor-F[1]), *precision*, y *recall* (exhaustividad) para medir sus resultados en las dos bases de datos que usaron para sus pruebas (estos términos de estadística serán definidos posteriormente).

Los valores máximos obtenidos en los entrenamientos con los conjuntos de entrenamiento *Oxford* y *Tuscaloosa*, para la clase de barrera metálica son los siguientes:

1. Precisión: 0.93
   (a) Conjunto: *Oxford*
   (b) Clasificador: *CNN-separateLSTM*
2. Exhaustividad: 0.86
   (a) Conjuntos: *Oxford* y *Tuscaloosa*
   (b) Clasificadores: *CNN-RF* y *CNN-DT*

[1]De acuerdo con colaboradores de Wikipedia (2021) el valor-f se considera una media armónica que combina el valor de precisión y exhaustividad.

3. Valor-F: 0.88

   (a) Conjunto: *Tuscaloosa*

   (b) Clasificador: *CNN-separateLSTM*

Los modelos *CNN-separate LSTM*, *CNN-RF* y *CNN-DT*, mencionados anteriormente, son todos RNC, en el primero, '*separate LSTM*' hace referencia a una red recursiva de *Long Short Term Memory*[2] que actúa por separado; en el segundo, '*RF*' se refiere a *Random Forest* [3]; y en el tercero, '*DT*' es *Decision Tree*[4]. En todos los casos, los rasgos extraídos por la RNC se procesan al final con el método mencionado y en la figura 4.1 podemos ver sus resultados para las barreras metálicas.

El equipo utilizado fue el siguiente: *Intel Xeon* CPU E5-2687w v4@3.00 GHz con 64 GB de memoria principal y una tarjeta gráfica *Nvidia Quadro* K6000 GPU con 2,880 núcleos y 12 GB de memoria.

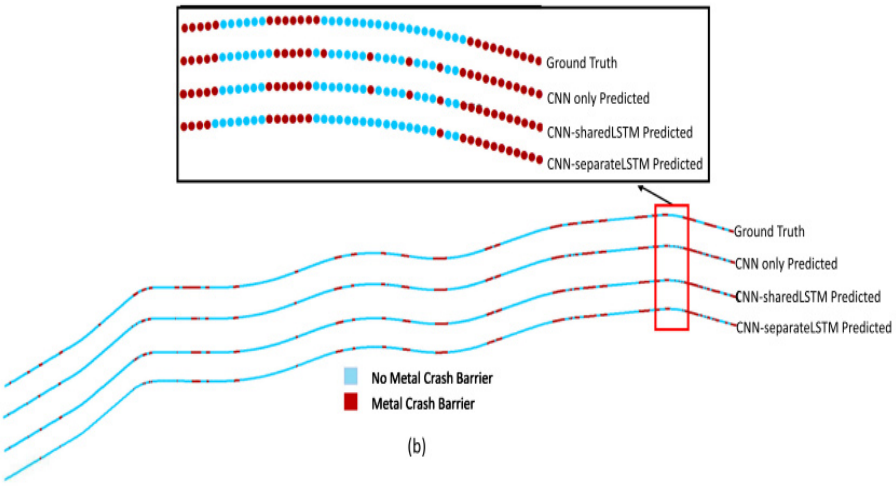


Figura 4.1: Mapas de resultados con el conjunto de prueba set_oxford de Sainju, A. M. and Jiang, Z. (2020) para cada categoría

Otra investigación del tema es la de Graf, S. et al. (2019), donde desarrollan un método para georreferenciar la presencia de barreras en las orillas de carreteras en Alemania. El objetivo principal era crear un registro que se utilizaría para analizar la seguridad de la red vial (el de colisiones de vehículos con vida salvaje); la idea fue evitar hacer esta geo-referenciación a mano.

[2]En Géron (2019, pág . 515) el autor menciona que la clave de estas redes está en aprender que guardar en el estado de largo plazo, que desechar y que quedarse de esto.

[3]En Géron (2019, pág . 189) se menciona que estos son un conjunto de *Decision Trees* en donde la predicción se realiza tomando en cuenta el total de votos que tenga la categoría (un voto es un clasificador diciendo "lo que entró pertenece a la clase A").

[4]En Géron (2019, pág . 175) se menciona que esto es un algoritmo que puede realizar clasificación, regresión e incluso tareas de múltiples salidas.

Graf, S. et al. (2019) utilizaron la red preentrenada *Inception* V3 (esta red es una RNC). El resultado de este estudio para la categoría de barreras, en palabras de los autores, fue el siguiente. "*Over 92 % of the images were classified correctly by the DNN*", esto se traduce a que se clasificaron correctamente más del 92 % de las barreras con la DNN [5] (*Deep Neural Network*). Podemos ver en la figura 4.2 una representación gráfica de sus resultados.

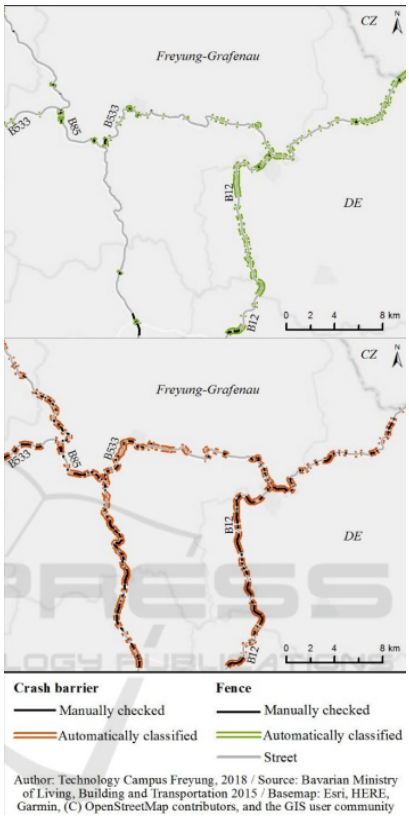


Figura 4.2: Mapeo resultante de registro manual y de forma automática de Graf, S. et al. (2019)

Por último, tenemos la investigación de Rezapour, M. and Ksaibati, K. (2021) donde comparan dos métodos para abordar la gestión de recursos. Uno de dichos procesos es la recolección de características de barreras en la orilla de carreteras. Los métodos utilizados incluyeron *transfer learning* (aprendizaje por transferencia) y *non-transfer learning* (aprendizaje sin transferencia).

Rezapour, M. and Ksaibati, K. (2021) emplearon las redes preentrenadas *inception v3*, *denseNet 121*, y *VGG 19*. Los autores mencionan lo siguiente sobre sus resultados "*This study achieved an accuracy of 97 % by the VGG 19 network*", esto se traduce a que se logró una exactitud del 97 % con la red *VGG 19*. Mencionan también que para esta red se empleó aprendizaje por transferencia, la red que no se basaba en este

[5] Los autores utilizan la abreviatura *DNN* la cual se refiere a una red neuronal profunda, pero utilizaron una RNC y utilizaron aprendizaje por transferencia.

método tuvo una exactitud del 85 %; sin embargo, fue mejor que las otras dos en otros rubros (*inception* y *denseNet*).
En la figura 4.3 vemos gráficamente los resultados del entrenamiento para los modelos que Rezapour, M. and Ksaibati, K. (2021) utilizaron en su investigación. La gráfica superior del par representa la pérdida en el entrenamiento (linea azul) y la validación (linea verde), la gráfica inferior muestra la métrica de exactitud en el entrenamiento (linea azul) y la validación (linea verde).

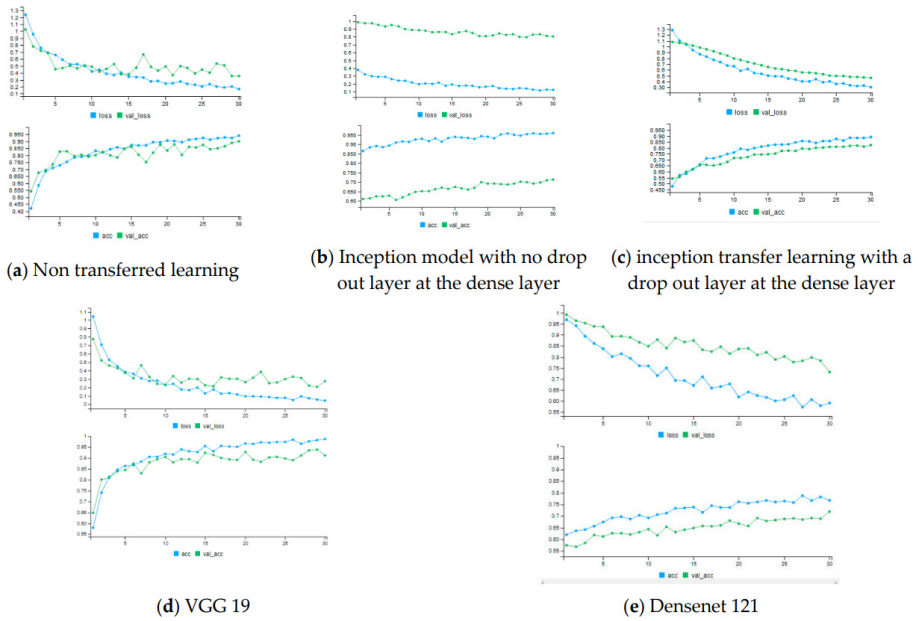


Figura 4.3: Gráficas de entrenamiento y validación de los modelos en la investigación de Rezapour, M. and Ksaibati, K. (2021)

Como hemos visto, se han logrado avances sobre la detección de barreras metálicas, abordando diferentes situaciones. Esto demuestra la utilidad de estos sistemas. Sin embargo, los resultados reportados en estos trabajos sugieren que aún hay lugar para mejoras en futuras investigaciones.

Otro aspecto notable es que se trata de un tema reciente de interés en la comunidad científica. El uso de herramientas desarrolladas para la visión por computadora para optimizar los procesos de instituciones dedicadas al manejo de la red carretera.

Un sistema que pueda desempeñar esta tarea aparentemente simple es útil no solo dentro del área de la Ingeniería Civil en el tema de carreteras (específicamente en la seguridad vial), sino también en la Ingeniería Ambiental, la Geomática y la industria automovilística. A continuación se presentan algunos comentarios de los autores de los artículos presentados.

En su artículo del 2020 Sainju, A. M. and Jiang, Z. (2020) reportaron ser los primeros en explorar un enfoque de aprendizaje profundo con imágenes de *Google Street View*

para el mapeo de rasgos de seguridad vial.

Rezapour, M. and Ksaibati, K. (2021), mencionan la siguiente idea sobre el futuro del método que emplearon en su artículo: '*the proposed approach might be used to help drivers with object detection in autonomous vehicles even with a very low frequency of images*'. Lo anterior se traduce: 'el enfoque propuesto podría ser utilizado para ayudar a conductores en vehículos autónomos con detección de objetos con una frecuencia muy baja de imágenes'.

Graf, S. et al. (2019), mencionan lo siguiente sobre su trabajo: 'la metodología proporcionó el material relevante para analizar colisiones de vehículos con vida salvaje, cuando al principio no había datos disponibles de estas infraestructuras en la orilla de la carretera'.

Ahora en el siguiente trabajo de investigación como en los mencionados anteriormente, se muestra un ejemplo de la aplicación de técnicas del aprendizaje de máquina a problemas de ingeniería.

Capítulo 5

Marco teórico

5.1 Introducción

Comenzaré hablando sobre el aprendizaje de máquina, tema al cual pertenece el sistema usado en esta investigación. Los métodos del aprendizaje de máquina no son comunes en el repertorio de herramientas para resolver problemas de ingeniería civil, pero algunos de ellos podrían ser una buena alternativa para la solución de problemas en esta área.

Si bien para programar un sistema de esta naturaleza no se requiere un entendimiento profundo del tema, lo mejor antes de usar cualquier herramienta es entender para qué sirve, qué podemos hacer con ella y cómo usarla.

La disciplina de donde viene el método es el aprendizaje de máquina que, de acuerdo con la referencia Géron (2019, pág . 2), puede ser definida como: "*The science (and art) of programming computers so they can learn from data*". Se menciona también una definición más formal de Arthur Samuel 1959, "*Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed*". En pocas palabras, el aprendizaje de máquina le permite a las computadoras aprender sin necesidad de programar explícitamente lo que deben saber.

Desarrollando un ejemplo de lo que significa programar explícitamente, que es familiar para un estudiante de ingeniería civil. Consideremos el diseño de un filtro de spam. El spam son correos no deseados, como las promociones de tiendas o de compañías de software.

Una forma de resolver el problema sería identificar palabras que siempre se presenten dentro de estos correos. Sin embargo, como lo menciona Géron (2019, pág. 3), si empleáramos la programación convencional (explícita), obtendríamos como resultado una larga lista de reglas que lograrían nuestro objetivo. No obstante, con un programa así, sería difícil darle mantenimiento. Además, habría que incluir manualmente las condiciones que definen lo que es el spam, lo que podría resultar en la eliminación de un correo que no lo es al actualizar la definición.

Si usáramos técnicas de aprendizaje de máquina (programación no explícita), nuestro programa aprendería los patrones que definen el spam y podría volverse un sistema robusto con el tiempo. Es decir, que él solo iría actualizando su definición de spam.

Otra razón por la cual usar aprendizaje de máquina es mencionada por Géron (2019, pág. 4); que sirve para problemas sin algoritmos conocidos que los resuelvan, o para los que resultan algoritmos complejos con los métodos convencionales.

Tomando en cuenta lo anterior, podemos comprender el motivo de emplear una de las técnicas de aprendizaje de máquina para resolver el problema central de esta investigación; necesitamos poder adaptarnos a cualquier carretera, independientemente de las condiciones climáticas o geográficas de la región. Además, no tendríamos que ingresar manualmente las condiciones que definen el elemento a clasificar, sino que el programa las aprendería de forma autónoma.

5.2 Redes neuronales artificiales

Antes de adentrarnos a las Redes Neuronales Convolucionales, debemos entender un concepto más general: las redes neuronales artificiales. Necesitamos comprender cuándo surgieron, de dónde proceden y qué funciones desempeñan.

Y antes de iniciar con la historia, hablaremos acerca de lo que es una red neuronal artificial. Se puede entender como un algoritmo al que le proporcionamos muchos ejemplos de cosas que queremos que identifique durante el proceso de entrenamiento. Durante el entrenamiento, medimos qué tan bueno es el sistema en la tarea de identificar la información deseada. Luego, lo ponemos a prueba con información que nunca antes ha visto y volvemos a medir qué tan bien hace su trabajo. A esta última etapa de pruebas se le conoce como validación. En el entrenamiento evaluamos qué tan buena es la red, y en la validación observamos si sigue desempeñándose bien frente a nueva información o no.

Los sistemas de redes neuronales artificiales, según Géron (2019, pág . 280), fueron introducidos en 1943 por el neurofisiólogo Warren McCulloch y el matemático Walter Pitts en su artículo de investigación titulado "*A Logical Calculus of Ideas Immanent in Nervous Activity*". En este trabajo, presentaron un modelo computacional simplificado de cómo podrían funcionar las neuronas biológicas en los cerebros de animales para realizar operaciones empleando lógica proposicional. Esta es considerada como la primera arquitectura de una red neuronal artificial.

En aquel entonces, estos sistemas tuvieron un gran auge pero de acuerdo con Géron (2019, pág . 280), esta tendencia no perduró. En los 80s hubo un resurgimiento de interés en investigar este tema, y en los 90s, cuando se habían desarrollado técnicas más poderosas, se volvió a dejar a un lado el estudio de las redes neuronales artificiales.

Actualmente, hay un fuerte interés en el estudio de estos temas. Esto se debe, en parte, a los resultados sorprendentes que estas metodologías han mostrado en diversos ámbitos. Por otro lado, está el deseo de las empresas de hacer uso de estas herramientas como parte de sus modelos de negocio. Todo esto ha sido facilitado por el gran incremento en el poder computacional, haciendo posible lo que hace aproximadamente 30 años no lo era.

La historia de las redes neuronales artificiales está ligada al sistema biológico de donde provino la inspiración para estos modelos; por lo tanto es relevante describir este mecanismo natural.

Una neurona, como la mostrada en la figura 5.1, es una célula que tiene en su cuerpo un núcleo y otros componentes complejos. Posee varias ramificaciones que se extienden lejos de su cuerpo, llamadas “dendritas", así como una ramificación de longitud variable denominada “axón". En un extremo de este último componente hay varias ramificaciones denominadas “telodendrones", y en las puntas de estas, hay estructuras microscópicas denominadas “terminales sinápticas", estas están conectadas a las dendritas o los cuerpos celulares de otras neuronas.

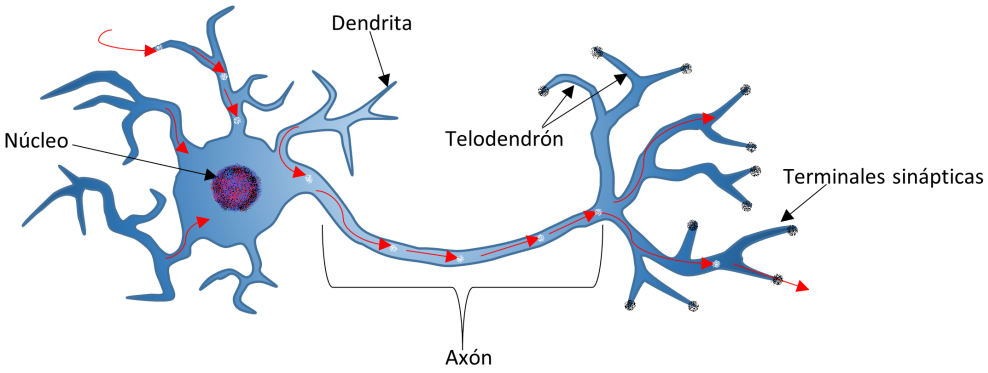


Figura 5.1: Neurona biológica y algunas de sus partes

Las neuronas producen impulsos eléctricos denominados potenciales de acción, que viajan por el axón y hacen que la sinapsis libere neurotransmisores (en la figura se muestra con la flecha roja el recorrido de tal impulso a través de la neurona). Cuando una neurona recibe una cantidad suficiente de estos, puede liberar su propio impulso eléctrico, pero esto depende de los neurotransmisores; entonces, existe la probabilidad de que se libere o no el impulso eléctrico.

Siendo específicos y de acuerdo con Landau (1998, pág. 6-7), la membrana celular mantiene una concentración dentro y fuera de la célula de varios iones [1] haciendo uso de bombas de iones y canales controlables de iones. Cuando la neurona esta en reposo, los canales están cerrados. Debido a varios factores, el interior de la neurona tiene un

[1] Los principales de acuerdo con (Landau, 1998, pag. 6) son: Na^+ (sodio), K^+ (potasio) y Cl^- (cloruro).

potencial eléctrico negativo comparado con el fluido exterior.

Una excitación eléctrica local suficientemente fuerte disminuye el potencial negativo temporalmente, lo que abre canales específicos de iones. Esto provoca una reacción en cadena de canales abriéndose y cerrándose, lo cual genera un impulso eléctrico breve que se propaga a través de la membrana rápidamente (este es el potencial de acción).

Tomando en cuenta lo presentado por Landau (1998, pág. 7), el potencial de acción sirve como una señal de comunicación, propagándose y bifurcándose a través del canal de salida de una neurona (axón) a otra. La unión entre la salida de una neurona y la entrada de otra (dendrita) se denomina sinapsis; la llegada del potencial de acción a la sinapsis libera un neurotransmisor químico, que actúa selectivamente para abrir un canal de iones. Si este canal es de Na^{+}, incrementa el potencial en la sinapsis receptora, lo cual aumenta la posibilidad de que esa neurona se active (esta sinapsis se denomina excitatoria), si es Cl^{-} el potencial se reduce y del mismo modo la probabilidad de activación (esta sinapsis se denomina inhibidora).

La activación de una neurona depende del efecto acumulativo de todas las señales excitatorias e inhibidoras, también existe la posibilidad de que la llegada del potencial de acción no libere el neurotransmisor. En las palabras de Landau (1998) "*This introduces an element of uncertainty, or noise, into the operation of the machinery*", que se traduce como "Esto introduce un elemento de incertidumbre, o ruido, en la operación de la maquinaria".

Pasando a la neurona artificial, las cuestiones que se preservan del mecanismo natural son las siguientes:

1. Transmisión de información de una neurona a otra.
2. El elemento de incertidumbre o ruido.
3. El efecto acumulativo que propicia la activación de una neurona.

En el primer punto hay una consideración, la información será transformada antes de transmitirla a otra neurona en la red (la transformaremos con una serie de productos y una suma). El segundo punto se refiere al hecho de que partes de la red carecen de interpretabilidad; aunque sabemos qué ocurre en ellas no podemos interpretar los resultados que obtenemos. El tercer punto se encuentra en como antes de transmitir la información a otra neurona, primero sumamos todas las contribuciones de las conexiones a la neurona que transmite los datos.

Ahora comenzaremos a hablar de modelos que toman como bloque básico la neurona artificial, con lo que surgió de la investigación realizada por McCulloch y Pitts. La neurona artificial, según Géron (2019, pág . 282), consta de una o más entradas binarias "On/Off", que produce una salida cuando más de cierto número de sus entradas están activadas. En su investigación mostraron que con un modelo tan simplificado,

era posible crear una red de neuronas artificiales que calcula cualquier proposición lógica que desees. Ejemplos de estas proposiciones son los operadores lógicos [2][3], que nos permiten simular sistemas complejos como lo podemos hacer con un lenguaje de programación convencional.

[2]Los operadores lógicos según IBM (2021b) “comparan valores booleanos".

[3]Los valores booleanos según IBM (2021a) representan “un valor de verdad; es decir, TRUE o FALSE".

Otro desarrollo importante para el tema a tratar, fue la creación de la arquitectura del perceptrón. De acuerdo con Géron (2019, pág . 284) este sistema sencillo está basado en una neurona artificial un poco diferente, denominada "*Threshold Logic Unit*" o "TLU", en español sería una Unidad Lógica de Umbral la cual se muestra en la figura 5.2[4]. En este sistema las entradas y salidas son números, y cada conexión de entrada está asociada a un valor de peso (w). Este modelo calcula la suma ponderada de sus entradas para luego aplicar una función a este valor.

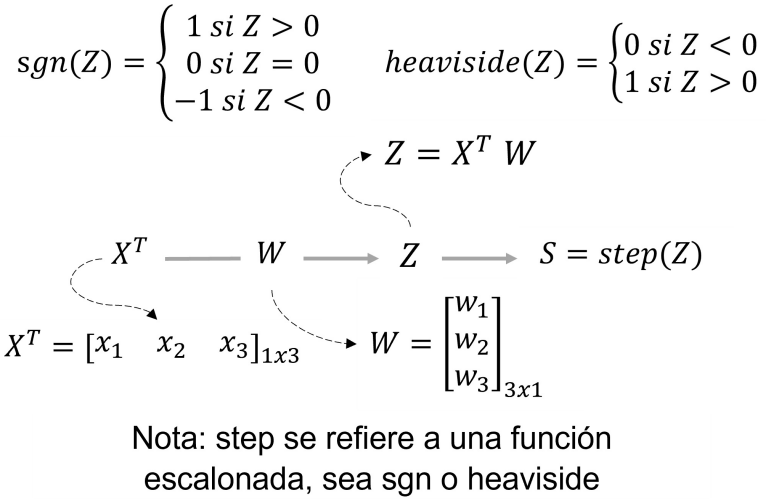


Figura 5.2: Unidad lógica de umbral y sus componentes.

[4]Sgn o heaviside son dos funciones escalonadas (*step functions*) comunes que se utilizan para calcular la salida de la *"TLU"*.

El perceptrón, como lo describe Géron (2019, pág . 285), está compuesto por una capa de TLUs, cada TLU está conectada con cada entrada. El término "totalmente conectada"se refiere a que cada neurona en una capa está conectada a todas las neuronas de la capa anterior. Se incluye además un término denominado sesgo o "*bias*" en inglés, que se integra al sistema como una neurona que envía una señal de valor numérico constante.

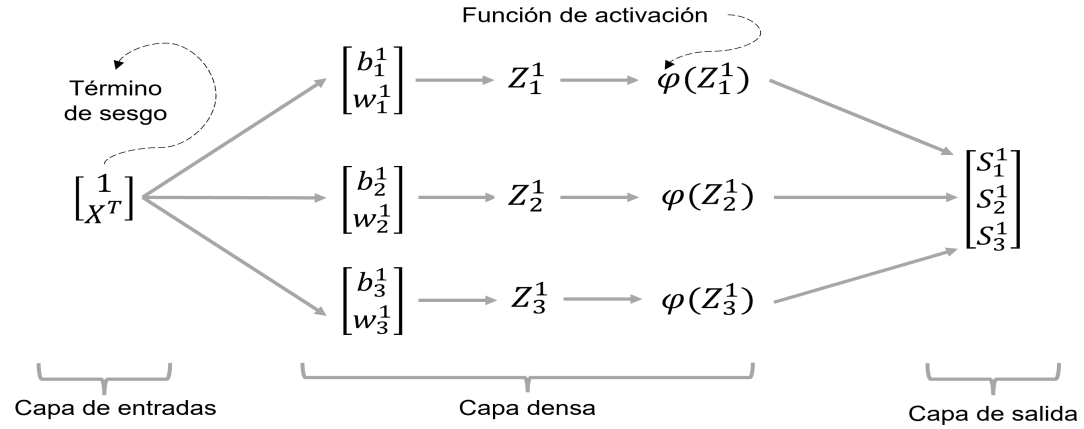


Figura 5.3: Perceptrón de tres neuronas.

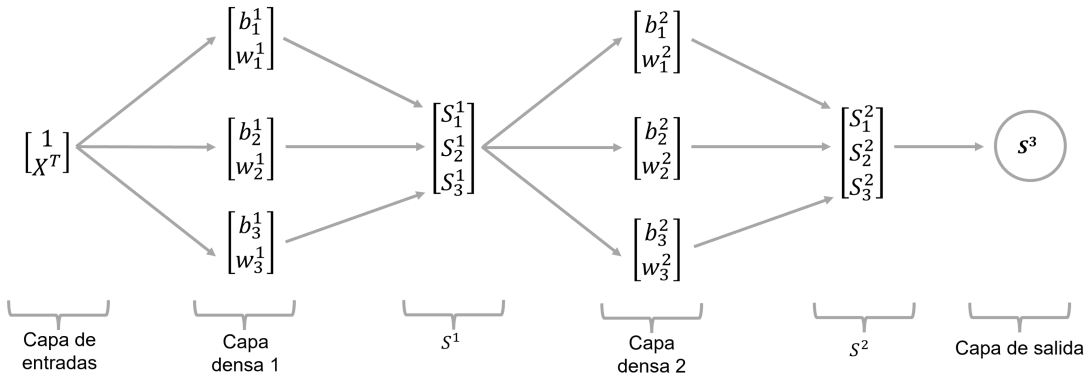
En resumen los componentes de estos sistemas, con base en lo expuesto por Géron (2019, pág . 286) son:

1. Entradas
   (a) Valores numéricos a procesar.
2. Matriz W
   (a) Contiene los pesos.
   (b) Una fila por neurona de entrada.
   (c) Una columna por neurona en la capa.
3. Vector b o "*bias*" (sesgo)
   (a) Posee un término por neurona.
4. Función de activación
   (a) Aquí ingresa la suma ponderada Z.
   (b) En las TLUs se denomina "*Step function*" o "Función de paso".

Entonces, ¿Qué es una red neuronal artificial? Es una colección de neuronas artificiales organizadas en capas que desempeñan una función específica de procesamiento de datos. En estos sistemas, los pesos en la matriz W y los valores de **sesgo (bias)** en el vector b son las **incógnitas** que encontraremos a través del proceso de **entrenamiento**, cuyo objetivo es el de lograr que nuestra red cometa menos errores al inferir sobre la información de entrada.

En esta investigación, la información de entrada son imágenes, y queremos una red cuyas capas nos permitan determinar si hay o no una barrera metálica en la imagen de entrada en términos de probabilidad. Es decir, si la red nos devuelve un 0.9 como resultado, significa que hay un 90 % de probabilidad de que haya una barrera en la imagen procesada.

A continuación, explicaré cómo funciona el procesamiento de datos en una red totalmente conectada, comenzando por su diagrama básico:



Nota: el superindice se refiere a la capa, se han expandido vectores por propósitos ilustrativos

Figura 5.4: Red totalmente conectada

Podemos observar que hay tres tipos de capas principales: la de entradas, donde en el caso del sistema diseñado tendremos valores de píxeles; las capas ocultas o densas, donde se procesa la mayor cantidad de datos; y la capa de salida, donde se expresa la probabilidad de clase. Ahora, el procesamiento de la información se muestra de forma matemática en las siguientes figuras (5.5, 5.6, 5.7).

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix}_{nx1}$$

Figura 5.5: Capa de entradas como vector

En el sistema diseñado, lo que ingresa a la red totalmente conectada es una lista de valores de píxel, que se ve representada como el vector [5] X en la figura 5.5.

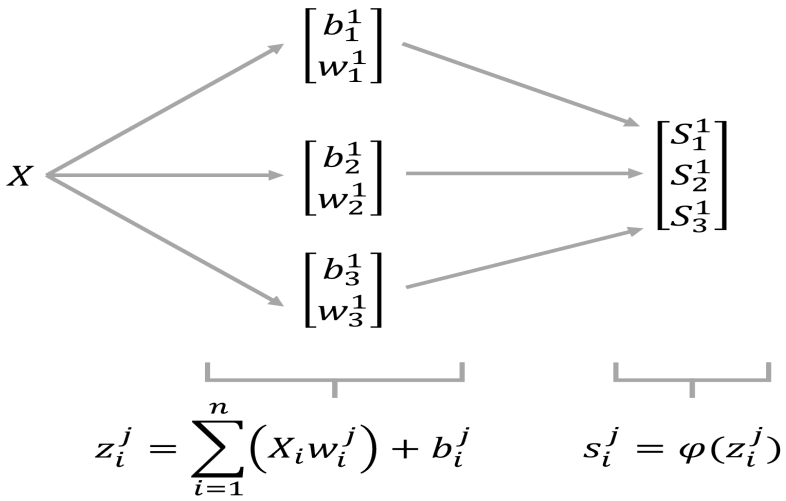


Figura 5.6: Capas densas y ecuación central

En la figura 5.6, tenemos las capas densas, donde se realiza la transformación de los datos de entrada. Cada elemento de la capa de entrada está conectado a todas las neuronas en la primera capa densa. Esto significa que la primera neurona recibirá todos los datos de entrada, al igual que la segunda neurona también. La diferencia radica en los pesos asociados a cada conexión de un dato de entrada con la neurona a la que se conecta.

Para transmitir un dato z_i de la primera neurona a la siguiente capa, seguimos los pasos siguientes:

1. Realizaremos el producto de todos los datos de entrada [6] x_i que llegan a la primera neurona con sus pesos de conexión correspondientes w_i, y los sumaremos como se muestra en la ecuación de z_i.

2. Sumaremos a este resultado el valor de sesgo (b_i o *bias*) asociado a la primera neurona.

3. Aplicamos a la suma de valores de z_i (resultados de los datos conectados a la neurona analizada) una función φ que nosotros hemos seleccionado previamente.

Este proceso se repite para todas las neuronas de la capa actual, y posteriormente se repite todo el proceso con el resto de las capas (excepto la de salidas).

[5] Las imágenes son un arreglo bidimensional de valores de píxel. En el caso de las imágenes a color son un volumen compuesto por tres arreglos bidimensionales apilados uno sobre otro. Estos arreglos representan los canales de color: uno para los tonos azules, otro para los verdes y un tercero para los rojos.

[6] Al reorganizar los elementos de una matriz, esta se expresa como un vector.

En la capa de salidas obtendremos la probabilidad de que en la imagen de entrada haya una barrera metálica, esto se determinará aplicando una función ε a la cual entrará la salida de todas las neuronas en la capa previa, estos valores se sumarán y el resultado de la función de salida será un valor entre cero y uno como se muestra en la siguiente figura:

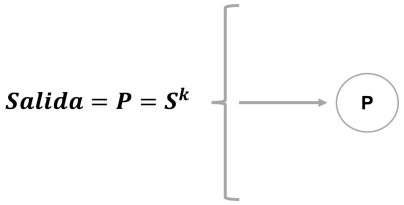


Figura 5.7: Capa de salida

Con el desarrollo del perceptrón también surgió una analogía con la regla de Hebb. Donald Hebb, en su libro "*The organization of behaviour*" de 1949, sugiere que cuando una neurona biológica desencadena a otra de forma recurrente, la conexión entre ambas se vuelve más fuerte. En la analogía númerica el refuerzo de conexiones en las redes neuronales artificiales se logra al minimizar el error [7] de la red al hacer predicciones, modificando los valores de pesos y sesgos de la red.

El proceso a través de cual se refuerzan las conexiones depende del algoritmo de "*backpropagation*" o propagación hacia atrás, el cual de acuerdo con Olah (2014a) "*Is the key algorithm that makes training deep models computationally tractable*", es decir el algoritmo clave que hace el entrenar modelos profundos computacionalmente manejable, pues al usarlo con el descenso por gradiente[8] hace que un proceso largo de entrenamiento dure semanas y no años.

Fundamentalmente, como menciona Olah (2014a), la propagación hacia atrás es una técnica para calcular derivadas de forma rápida. El punto clave es que, a diferencia de la diferenciación convencional o "hacia adelante", la propagación hacia atrás nos proporciona todas las derivadas de las entradas de una expresión con respecto a una salida (es decir, cómo se ven afectadas las entradas por las salidas de una función). En cambio, en el otro caso, obtenemos las derivadas de las salidas con relación a las entradas (es decir, cómo se ven afectadas las salidas por las entradas de la función).

Es difícil ver cómo esto es conveniente en un problema sencillo de pocas variables, pero cuando se trabaja con un gran número de variables, como es el caso del entrenamiento de una red neuronal artificial, se acelera múltiples veces el proceso de obtener derivadas, las cuales se usan para minimizar el error de nuestra red durante las predicciones.

Se han explicado las partes del sistema y como se procesa la información, pero no se ha aclarado cómo es que estos logran clasificar exitosamente las imágenes, de las cuales

[7] El error de forma simple es la diferencia entre el valor real de la variable y la predicción de la red, la definición matemática de este concepto se presenta en la Sección 5.6.

[8] El descenso por gradiente de acuerdo con Khan Academy (2022) es: "Un algoritmo que estima numéricamente dónde una función genera sus valores más bajos".

únicamente entran pixeles y salen predicciones. Por lo que en el anexo C podemos encontrar una explicación detallada de este interesante proceso.

En la siguiente sección, se describirá el sistema utilizado en esta investigación. Este sistema incorpora una red totalmente conectada, la cual se describe en este apartado. Además se utiliza la ecuación mencionada en este capítulo, la cual se presenta tanto en la red totalmente conectada como en la operación de convolución, de donde surge el nombre de estos modelos, que realiza la red para aprender rasgos de las imágenes.

5.3 Redes neuronales convolucionales

En este capítulo hablaremos sobre la historia del sistema central de esta investigación; de dónde surgió la idea y cómo funciona.

Según Géron (2019, pág. 445), las Redes Neuronales Convolucionales (RNC o CNN) emergieron del estudio de la corteza visual y se han usado en el reconocimiento de imágenes desde los años 80. Hoy en día, son empleadas en vehículos autónomos, servicios de búsqueda de imágenes, sistemas de clasificación automática de videos y más.

Algunas aplicaciones generales de las RNCs [9] son las siguientes:

1. Verificación de firmas.
2. Predicción del valor de acciones en la bolsa.
3. Detección de tumores cerebrales usando imágenes de resonancias mágneticas.

Algunas aplicaciones en la ingeniería civil[10] son las siguientes:

1. Clasificación automática de cobertura de suelo y zonas potenciales de acuiferos.
2. Detección de grietas en estructuras de concreto.
3. Detección de daño en estructuras de acero.

La historia de estos sistemas, según Géron (2019, pág. 446), comenzó cuando David H. Hubel y Torsten Wiesel realizaron experimentos con gatos en 1958 y 1959. Mostraron que muchas neuronas en la corteza visual tienen un campo receptivo local. Esto significa que solo reaccionan a estímulos visuales localizados en una región limitada del campo visual; estos campos pueden empalmarse y juntos abarcar todo el campo visual. Los autores mostraron también lo siguiente:

1. Algunas neuronas reaccionan solo ante patrones específicos (pueden compartir el campo receptivo, pero reaccionar ante patrones diferentes).
2. Algunas neuronas tienen campos receptivos más grandes y reaccionan a patrones que son combinación de patrones simples.

Esos experimentos inspiraron el neocognitrón, que fue introducido en 1980 y gradualmente evolucionó a lo que conocemos como una CNN. Un hito de este tema fue la investigación de 1998 “Gradient-Based Learning Applied to Document Recognition” de Yann LeCun et al, donde se introdujo la famosa arquitectura LeNet-5, ampliamente

[9] En orden de aparición las referencias generales son las siguientes: (Jain, V. et al., 2022), (Sarhan, A. M. , 2020) y (Sen, J. and Mehtab, S., 2020).

[10] En orden de aparicion las referencias son las siguientes: (Tegegne, A. M., 2022), (Ghazvineh, S. et al., 2021) y (Mohamad, A. et al., 2019).

usada por bancos para reconocer números de cheques escritos a mano. En esta arquitectura se introduce la capa convolucional y la capa de muestreo o '*pooling*', como se le nombra en inglés.

Las Redes Neuronales Convolucionales (como las diseñadas para nuestra aplicación de barreras F-mod y C-mod) siguen la siguente estructura, y al final la probabilidad determinada por la red aparece como el último nodo. La figura 5.8 ilustra las capas de la red, el nodo de salida indicado con la letra P y tres ejemplos de filtros convolucionales en la capa de convolución.

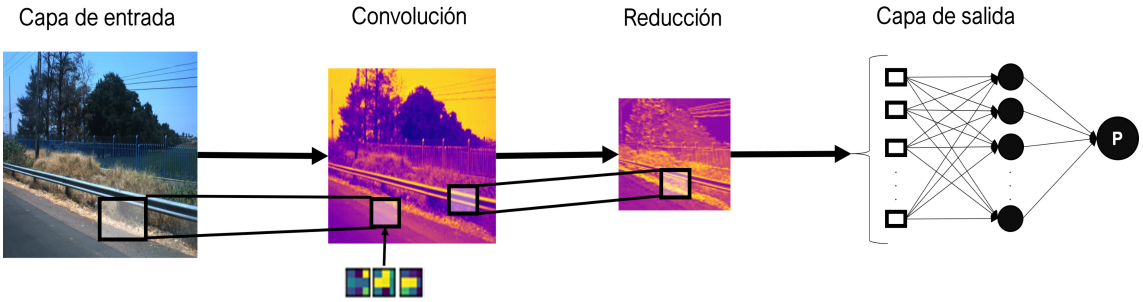


Figura 5.8: Componentes de una red neuronal convolucional

5.3.1 Capas convolucionales

Ahora, para adentrarnos en el sistema diseñado, hablaremos un poco sobre las capas y los componentes principales de estas redes, especializadas pero no restringidas al procesamiento de imágenes.

Las capas convolucionales hacen algo muy sencillo, escanean la imagen que entra a la capa con un filtro convolucional (kernel convolucional) de tamaño definido por el usuario. Este filtro es una matriz llena de pesos w aleatorios cuyo valor cambia con el entrenamiento. Durante este escaneo se determina el valor s_i de activación con la ecuación usada en la red totalmente conectada para cada neurona, mediante el producto entre los pesos del filtro y los valores de píxel x_i en la región analizada de la imagen.

El resultado de este escaneo de toda la imagen de entrada es una representación interna conocida como mapa de características, que muestra las regiones de la imagen, donde hubo mayor activación de neuronas como se muestra en la figura 5.9. Esto nos informa sobre de los rasgos resaltados por el filtro [11]convolucional.

[11]Según Géron (2019, pág. 460), durante el entrenamiento la red aprenderá filtros útiles para su tarea.

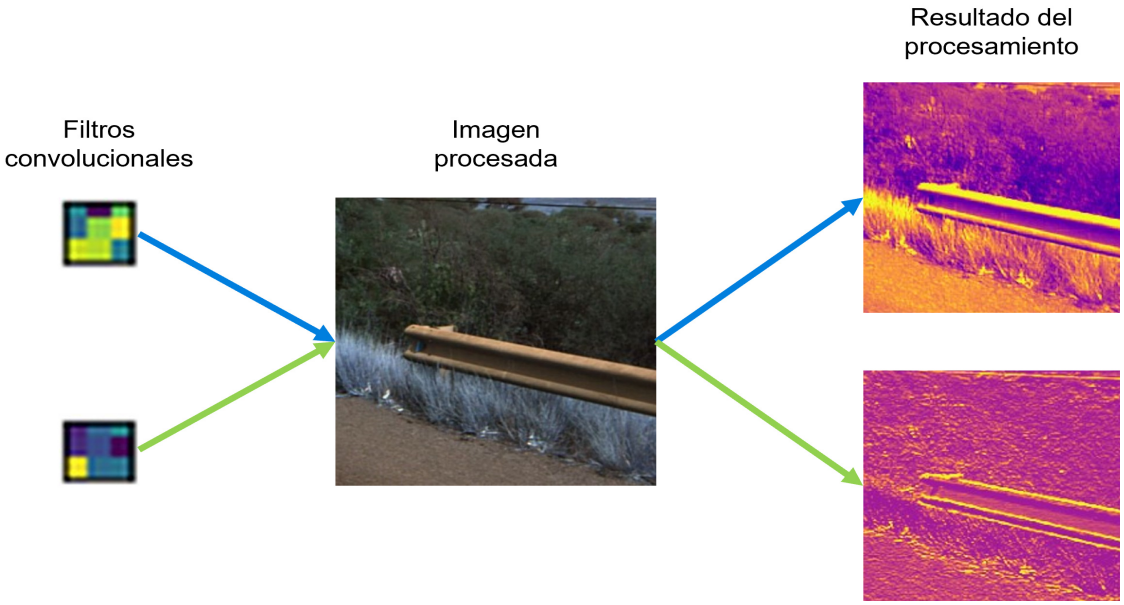


Figura 5.9: Proceso de convolución

Las neuronas en la primera capa convolucional están conectadas a todos los pixeles de su campo receptivo en la imagen de entrada. A medida que avanzamos en capas subsiguientes, las neuronas están conectadas a regiones rectangulares de la capa anterior, lo que permite que la red se enfoque en rasgos más grandes a medida que avanza en las capas. Esta estructura jerárquica es común en imágenes reales y ayuda a la red a aprender características complejas a partir de otras más simples como lo menciona Géron (2019, pág. 447).

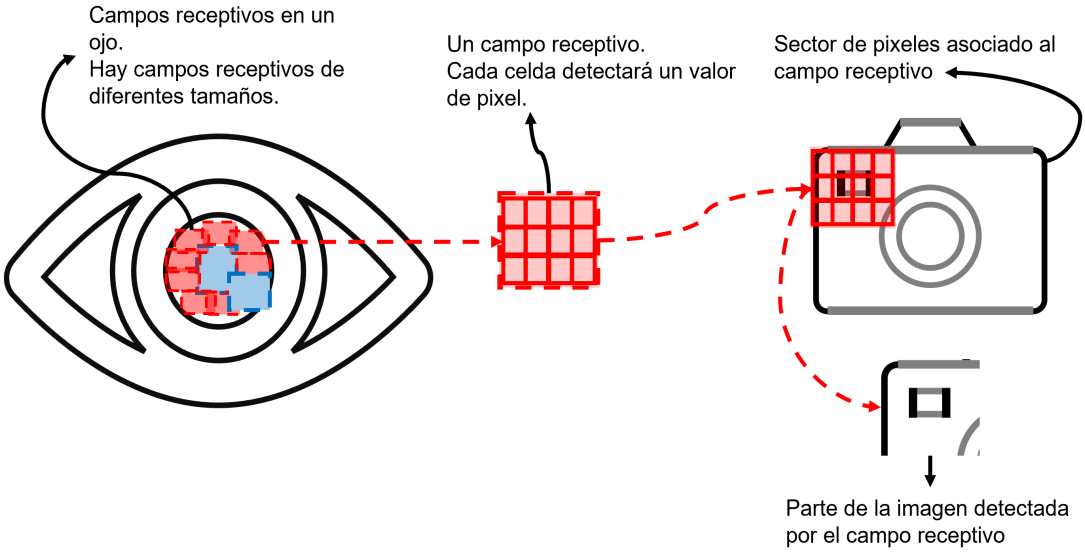


Figura 5.10: Campos receptivos locales.

Una manera sencilla de visualizar el conjunto de filtros es como un objeto 3D, donde

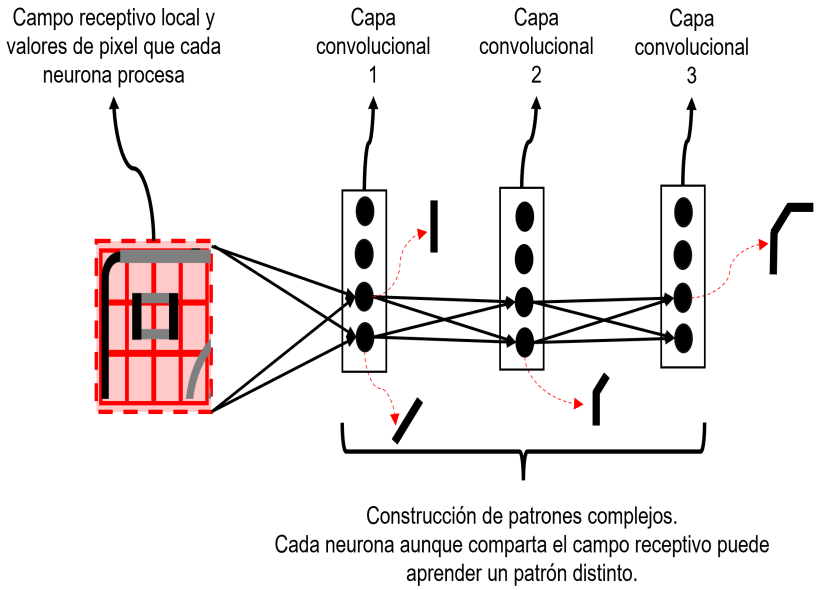


Figura 5.11: Jerarquía de patrones.

el volumen está compuesto por todos estos. Como resultado, se produce un volumen de mapas de características, con un mapa de características por filtro.

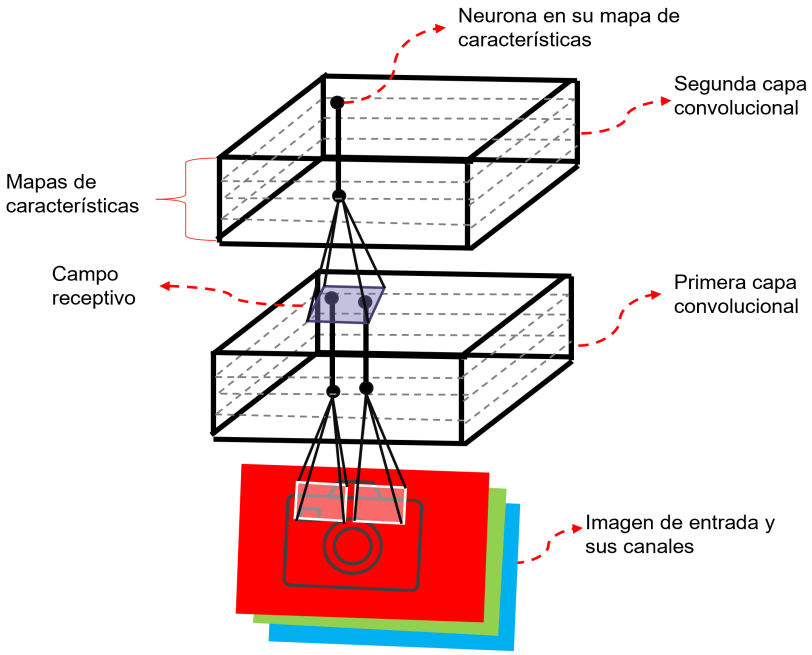


Figura 5.12: Sección de una red convolucional

5.3.2 Mapas de características

Los mapas de características, como se mencionó previamente, son lo que los filtros resaltan, es decir, muestran a qué reacciona la red. Esta representación interna ayuda a la red a aprender patrones, lo que le permitirá clasificar si los vuelve a encontrar en imágenes nuevas.

(a) Imagen de entrada

(b) Mapa de características

Figura 5.13: Efectos de la convolución

En los mapas de características, a cada píxel le corresponde una neurona, y cada una comparte parámetros (pesos y valor de sesgo) si están dentro del mismo mapa, como lo menciona Géron (2019, pág. 450). Los parámetros asignados a las neuronas en cada mapa son únicos. El campo receptivo de las neuronas se extiende a través de todos los mapas en capas previas, como se muestra en la figura 5.12.

5.3.3 Capas de pooling

Su objetivo, según Géron (2019, pág. 456), es el de submuestrear la imagen de entrada para reducir la carga computacional, el uso de memoria y el número de parámetros al procesar la imagen de entrada. Las neuronas de la capa de *pooling* escanean la imagen de entrada de manera similar a la convolución. Sin embargo, estas neuronas aplican una función específica, como tomar el valor máximo en la región analizada o el promedio de los valores. Esto tiene el efecto de reducir el tamaño de la imagen, lo que permite que la red aprenda rasgos más generales de la misma clase, mejorando su capacidad para distinguirla de otras.

5.4 Modelo empleado

Volviendo al aspecto técnico de la investigación, después de comprender la razón detrás del uso de una metodología de aprendizaje automático, el origen del tema y las operaciones que se realizan, definiremos el sistema desarrollado, que fue uno de aprendizaje supervisado. Como menciona Géron (2019, pág. 7), lo que distingue este enfoque de otros es que el usuario asigna etiquetas a las entradas del programa, por lo que desde el inicio conocemos cuáles tienen que ser los resultados de una clasificación.

Aplicaremos *batch learning*, también conocido como aprendizaje por lotes, tal como se menciona en Géron (2019, pág. 15). Esto implica que nuestro sistema no podrá aprender de manera incremental. En lugar de eso, se entrenará asumiendo que no necesitará actualizarse durante un período de tiempo. Los momentos designados para realizar estas actualizaciones dependerán de los cambios en los diseños de los elementos a clasificar, en este caso, las barreras metálicas en las orillas de las carreteras.

Un aspecto relevante para considerar en futuras mejoras del sistema es mencionado por Géron (2019, pág. 15): "*If you want a batch learning system to know about new data, you need to train a new version of the system from scratch on the full dataset*". Esto significa que si queremos que nuestro sistema de aprendizaje por lotes se adapte a nuevos datos, necesitamos entrenar una nueva versión del sistema desde cero utilizando un conjunto completo de datos nuevos.

5.5 Condiciones preliminares

En esta sección nos enfocaremos en los requisitos para diseñar nuestra CNN, comenzando por el más importante de todos: los datos. Estos no pueden ser simplemente seleccionados al azar en términos de cantidad y calidad; por lo tanto, es crucial considerar algunos puntos importantes. Como lo que dice Géron (2019, pág. 23) "*It takes a lot of data for most machine learning algorithms to work properly. Even for very simple problems you tipically need thousands of examples*". Esto significa que, para que funcionen adecuadamente las metodologías de aprendizaje automático, se requiere una gran cantidad de datos (imágenes), incluso miles de ellos, incluso para problemas sencillos. Afortunadamente, en el caso de este proyecto, contamos con una base de datos que cumple con este requisito en términos de tamaño.

Es crucial que la información empleada sea representativa, lo que significa que las imágenes seleccionadas deben proporcionar descripciones visuales precisas de los elementos a clasificar. Esto es importante para lograr una generalización efectiva en el contexto de las carreteras de un solo cuerpo. Como se menciona en Géron (2019, pág. 25), el proceso de muestreo para obtener estas imágenes debe tener en cuenta las situaciones reales a las que se enfrentará el programa en la práctica. Además, se debe procurar realizar un muestreo imparcial de la información siempre que sea posible para evitar sesgos en los datos.

Al diseñar desde cero, puede ser que la información disponible sea de mala calidad, esto hará que el sistema sea propenso a comportarse mal y rendir resultados no deseados. Si los datos están llenos de valores atípicos o ruido será difícil para el sistema detectar los patrones subyacentes; por lo tanto, es importante hacer una limpieza exhaustiva de las bases de datos.

La red neuronal necesita datos relevantes para aprender y realizar una clasificación precisa. Este proceso de filtrado de datos se conoce como "selección de características", como menciona Géron (2019, pág. 27). Implica elegir los aspectos más relevantes para la clasificación, simplificando así la tarea para el sistema. Como creadores, debemos entender claramente las características distintivas de los objetos para proporcionar a la red la información necesaria para aprender con precisión.

5.6 El entrenamiento

A partir de ahora nos centraremos en los aspectos técnicos del proceso mediante el cual desarrollaremos una RNC, similar a la empleada para abordar el problema central de la investigación.

En el entrenamiento, esencialmente lo que haremos será encontrar los mejores valores para nuestras incógnitas, que son los pesos w y sesgos b de la red. Esto se logrará a través de una función de costo, también llamada de pérdida, que determina el error en las predicciones, y un algoritmo optimizador que reducirá dicho error, proporcionándonos como resultado valores cuasi-óptimos para las incógnitas.

Lo que ocurre dentro de una sesión de entrenamiento de un sistema como el desarrollado es descrito en Géron (2019, pág, 290-291), y a continuación se enumeran los pasos:

1. Entra a la red un lote de imágenes, cuyo tamaño se denomina "batch size" en inglés. Este valor es elegido por el usuario.

2. Los píxeles de la imagen, que se encuentran en la capa de entrada, son procesados para llegar a las neuronas en la siguiente capa, como se mencionó en el capítulo de la red totalmente conectada y en el anexo C, el resultado s_j se pasa a la siguiente capa. Este cálculo se repite en todas las capas hasta llegar a la de salida. [12].

3. El algoritmo calcula el error de la predicción utilizando la función de pérdida (o "loss function" en inglés) para comparar la salida deseada con la real, y el resultado de esta comparación es el error.

4. Se calcula la contribución de cada conexión al error mediante la aplicación de la regla de la cadena [13] en la propagación hacia atrás, también conocida como "backpropagation" en inglés.

5. Se determina cuánto de estas contribuciones de error corresponde a las conexiones de la capa inferior, nuevamente utilizando propagación hacia atrás hasta llegar a la capa de entrada.

6. Una vez que se han calculado los gradientes de error de los pesos en las conexiones de todas las capas, el algoritmo realiza una operación de descenso de gradiente para ajustar todos los pesos de conexión en la red, utilizando los gradientes de error previamente calculados.

[12] A este proceso se le denomina "*forward pass*" o pasada hacia adelante, de acuerdo con Géron (2019, pág, 290) esto es similar a lo que ocurre al hacer predicciones, pero en este caso se preservan los valores para la pasada hacia atrás o el "*backward pass*"

[13] De acuerdo con colaboradores de Wikipedia (2021): "si z depende de una variable y y a su vez esta depende de x (esto es, z y y son variables dependientes) entonces z también depende de x, en tal caso la regla de la cadena enuncia que $\frac{dz}{dx} = \frac{dz}{dy} * \frac{dy}{dx}$„

En el anexo A se presenta el proceso de la propagación hacia atrás.

Un término que se utilizará a menudo para resumir todos estos pasos en uno solo es la palabra *epoch*. De acuerdo con Brownlee (2019), esto representa cuántas veces se han visto todos los datos de entrenamiento y se ha aplicado el algoritmo de descenso por gradiente. En la práctica, esto se hace por lotes, lo que significa que el algoritmo de descenso por gradiente se aplica solo después de haber visto un subconjunto de todos los datos de entrenamiento.

Otra parte del entrenamiento es el proceso de validación donde se pone a prueba la versión actual de la red entrenada con información que no ha visto antes. Esto nos daría una idea de cómo funcionaría nuestro sistema una vez implementado.

Una analogía presentada por Géron (2019) es que el descenso por gradiente es como estar en un lugar alto con neblina y querer bajar con cuidado. Uno buscaría el lugar con la mayor pendiente para descender, y una vez en este punto, repetiría el proceso de búsqueda del mejor sitio para bajar hasta llegar al fondo.

Numéricamente, esto significa medir el gradiente de la función de error en relación al vector de parámetros θ, y dirigirse en la dirección del gradiente descendente. Cuánto moverse en una dirección está determinado por el parámetro de tasa de aprendizaje o "*learning rate*" en inglés. Este es uno de los hiperparámetros que podemos modificar en nuestra red. Una vez que el gradiente sea cero, estaremos en un mínimo, es decir, el fondo.

En este ejemplo la función de pérdida es Error Cuadrático Medio (ECM). Esta función es la siguiente de acuerdo con colaboradores de Wikipedia (2020):

$$ECM = \frac{1}{n}\sum_{i=0}^{n}(\hat{Y}_i - Y_i)^2$$

Donde: $\hat{Y}$ es un vector de n predicciones y Y son los valores reales.

En las siguientes figuras se presenta el proceso y la función que describe el descenso por gradiente, de acuerdo con Géron (2019). Para una descripción detallada de donde surgen las ecuaciones, consultar el anexo B:

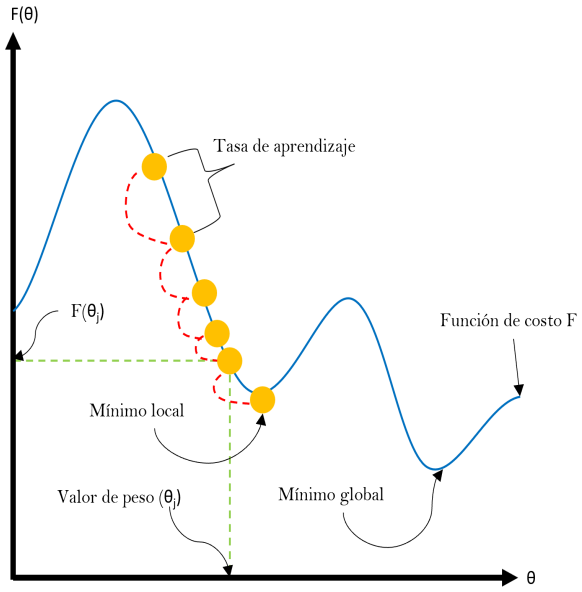


Figura 5.14: Proceso de optimización

Función de costo usada en la CNN: $$F = MSE = \frac{1}{m}\sum_{i=1}^{m}\left(\hat{Y}^i - y^{(i)}\right)^2$$

Predicciones de la red: $$\hat{Y} = \theta^T \boldsymbol{x}^{(i)}$$

Calculo del gradiente de error de un peso en las conexiones de la red: $$\frac{\partial}{\partial\theta_j} MSE(\theta) = \frac{2}{m}\sum_{i=1}^{m}\left(\theta^T x^{(i)} - y^{(i)}\right)x_j^{(i)}$$

Calculo de todos los gradientes: $$\nabla_\theta MSE(\theta) = \begin{pmatrix} \frac{\partial}{\partial\theta_1} MSE(\theta) \\ \frac{\partial}{\partial\theta_2} MSE(\theta) \\ \vdots \\ \frac{\partial}{\partial\theta_n} MSE(\theta) \end{pmatrix} = \frac{2}{m}\boldsymbol{X}^T(\boldsymbol{X\theta} - \boldsymbol{y})$$

Paso de actualización de los pesos: $$\theta^{(siguiente\ paso)} = \theta - \eta\nabla_\theta MSE(\theta)$$

Figura 5.15: Ecuaciones involucradas

La función de costo utilizada en la RNC diseñada fue "*binary cross entropy*", que estaba originalmente en el ejemplo presentado por Phan (2021). Géron (2019) menciona que

esta función de costo es usada frecuentemente para medir qué tan bien un conjunto de estimaciones de probabilidades de clases se compara con las clases objetivo.
La función binary cross entropy se presenta a continuación:

p : Distribución de valores reales
q: Distribución de predicciones
$\boldsymbol{\theta}$: Vector de pesos
$\boldsymbol{x}$: Vector de entradas
y_i: Valor real
$\hat{y}_i$: Predicción o probabilidad de clase
N: Cantidad de datos
$g(z)$: Función sigmoide

(a) Definiciones

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} H(p_i, q_i) = \frac{1}{N}\sum_{i=1}^{N} [p_i * \log q_i]$$

$$J(\theta) = -\frac{1}{N}\sum_{i=1}^{N} [y_i * \log(\hat{y}_i) + (1 - y_i) * \log(1 - \hat{y}_i)]$$

$$\hat{y}_i = g(z) = g(\boldsymbol{\theta} \cdot \boldsymbol{x}) = \frac{1}{1 + e^{-(\boldsymbol{\theta} \cdot \boldsymbol{x})}}$$

(b) Ecuación

Figura 5.16: Binary cross entropy

Otro aspecto que forma parte del diseño y del entrenamiento es la selección del optimizador, el algoritmo que se hará cargo de reducir el error de nuestra red. En esta investigación, la decisión de utilizar "*ADAM*" como optimizador vino de lo mencionado por Phan (2021), pero también por lo mencionado por sus desarrolladores Kingma, D.P. and Ba, J. (2014): "*Our method is designed to combine the advantages of two recently popular methods: AdaGrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp (Tieleman Hinton, 2012) which works well in on-line and non-stationary settings*". Lo anterior significa que este método está diseñado para combinar las ventajas de dos métodos populares, AdaGrad que funciona bien con gradientes dispersos y RMSProp que trabaja bien en ambientes en línea y no-estacionarios. Por lo tanto se modificó el optimizador empleado en el material de referencia.
Otras ventajas de "*ADAM*" de acuerdo con Kingma, D.P. and Ba, J. (2014) son las siguientes:

1. Implementación sencilla.
2. Es computacionalmente eficiente y tiene pocos requerimientos de memoria.
3. Los hiperparámetros son intuitivos y requieren poca afinación.
4. Es apropiado para problemas con muchos datos o parámetros.
5. El método calcula tasas de aprendizaje adaptativas individuales para diferentes parámetros.

Continuando con el tema del entrenamiento, una situación que suele presentarse es el sobreajuste, el cual es una causa común de un mal desempeño y poca precisión durante esta fase. En Géron (2019, pág. 27), se tiene la siguiente definición informal del término: "*It means that the model performs well on the training data, but it does not generalize well*", es decir, que nuestro modelo trabajaría bien con imágenes que se usaron para entrenarlo, pero no generaliza bien, es decir, que no se desempeña tan bien en imágenes que no ha visto antes.

La generalización es importante pues para tener un modelo que sea capaz de servir a gran escala, debe poder responder de forma satisfactoria en condiciones nuevas. Esta adaptabilidad es la que le da tanta ventaja a estos métodos de aprendizaje de máquina en relación con otras metodologías. Por lo tanto, es importante identificar cuándo se está presentando esta condición.

Ahora que mencionamos el término de sobreajuste, debemos hablar sobre cómo detectarlo. Géron (2019, pág. 25) nos dice que si tu modelo se equivoca más al generalizar que durante el entrenamiento, esto puede significar que se está sobreajustando al conjunto de datos de entrenamiento, es decir, los está memorizando.

También puede ocurrir el caso contrario, donde se tienen modelos muy sencillos que no logran captar la esencia del problema que se trata de resolver (esto se denomina subajuste). Por lo tanto, es importante tener bien claro qué clase de problema se está tratando de resolver y qué métodos se ajustan mejor al problema para darle solución.

Sobre medidas correctivas Géron (2019, pág. 28) menciona lo siguiente: "*Constraining a model to make it simpler and reduce the risk of overfitting is called regularization*", esto quiere decir que restringir al modelo, para hacerlo más simple reduce el riesgo del sobreajuste, y a este acto se le denomina regularización.

La cantidad de regularización que se empleará está en función de los hiperparámetros que se modifiquen antes de realizar el entrenamiento de la red. En consecuencia, podemos afinar estos valores constantes para producir mejores resultados, lo cual es un paso crucial en el entrenamiento de los modelos.

La forma de determinar el valor adecuado para un hiperparámetro dado es mencionada por Géron (2019, pág. 31) cuando dice: "*One option is to train 100 different models using 100 different values for this hyperparameter*". Esto significa que a través de pruebas y errores es como podemos determinar si tenemos una configuración de valores de hiperparámetros buena (más no óptima) para nuestro modelo.

En la red diseñada, se emplearon los métodos de regularización *Dropout* y *Spatial Dropout*. Según Keras (2022), la capa de *Dropout* anula de forma aleatoria el valor de algunas entradas con una frecuencia determinada por la tasa seleccionada durante cada paso del entrenamiento[14]. Las entradas que no se anulan se escalan por un factor de $\frac{1}{1-tasa}$ para mantener la suma de todas las entradas inalteradas.

Labach, A. et al. (2019) menciona lo siguiente acerca del método: "*This method proved effective for regularizing neural networks, enabling them to be trained for longer periods without overfitting and resulting in improved test accuracy.*". Esto significa que es un método eficaz que nos permite entrenar durante más tiempo sin sobreajuste, lo que resulta en una mejora en la exactitud durante las pruebas.

[14]Labach, A. et al. (2019) menciona que en el artículo de investigación original donde se propone el método, se recomienda una tasa de 0.2 para capas de entrada y 0.5 para capas ocultas.

Spatial Dropout, como se menciona en Team, K. (2022), ignora mapas de activación enteros si hay correlación con los píxeles de otros adyacentes, lo que significa que se queda únicamente con los mapas de activación (mapas de características) únicos, esto mejora la generalización de la red, según Tompson, J. et al. (2015). Además, en Team, K. (2022) también se recomienda usar *Spatial Dropout* en lugar de *Dropout*.

5.7 La evaluación

Una parte central en el desarrollo de un sistema es cómo se midieron los resultados de este en la actividad para la cual fue diseñado, lo cual se expone a continuación. Comenzaremos definiendo la notación utilizada en las ecuaciones:

1. "TP" son los verdaderos positivos (*true positives*), imágenes que sabemos que tienen barreras
2. "TN" son los verdaderos negativos (*true negatives*), imágenes que sabemos que no tienen barrera
3. "FP" son falsos positivos (*false positives*), imágenes que la red determinó que contenían una barrera pero no es verdad.
4. "FN" son falsos negativos(*false negatives*), imágenes donde existe una barrera pero la red determinó que no.

Las ecuaciones de precisión y exhaustividad, tal como se presentan en Géron (2019, pág. 91-92), están complementadas con un diagrama de ejemplo. Donde se muestran los resultados de un clasificador de ejemplo que comete algunos errores. En los diagramas, los triángulos representan imágenes sin barrera y los círculos imágenes con barrera. La inclusión de estos diagramas tiene como objetivo facilitar la interpretación de las ecuaciones.

La precisión se calcula con esta expresión:

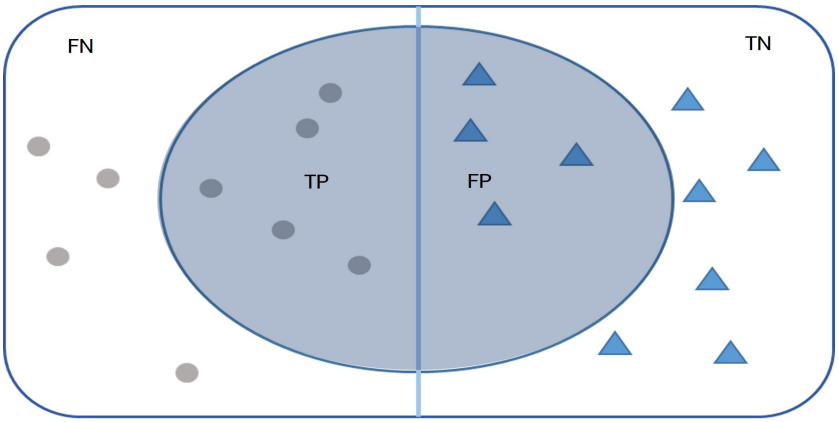


Figura 5.17: Espacio de medidas

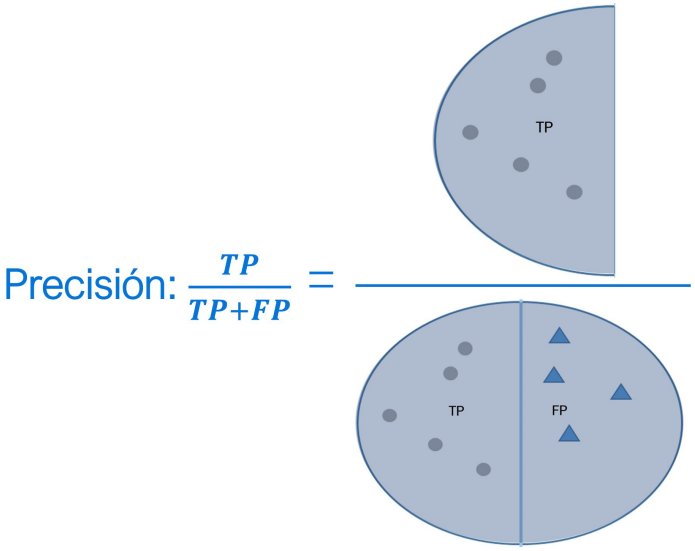


Figura 5.18: Ecuación

Una forma de entender este concepto es considerar cuántas veces, de todas las instancias en las que había una barrera en la imagen, nuestro modelo logró detectarla correctamente.

La exhaustividad, según la definición presentada por Purva (2020), es la medida de que tan bueno es nuestro modelo identificando correctamente positivos verdaderos. Se determina con la siguiente fórmula:

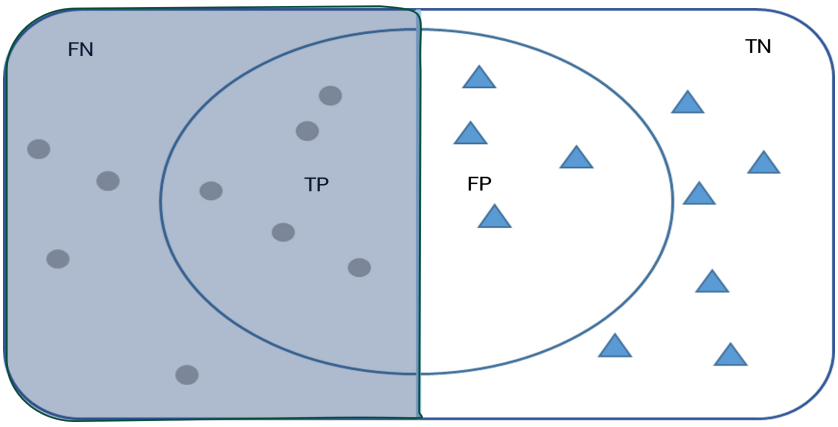


Figura 5.19: Espacio de medidas

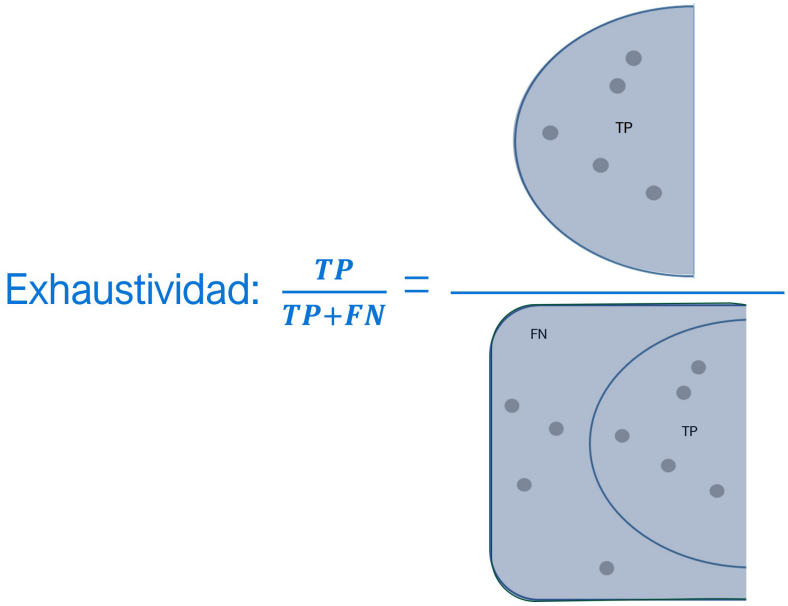


Figura 5.20: Ecuación

Otra métrica utilizada en la investigación es el Valor-F, que se incluyó debido a que comúnmente se reporta en investigaciones que involucran clasificadores, como el desarrollado en el presente trabajo de investigación.

El Valor-F, según Purva (2020), es una media armónica de la precisión y la exhaustividad, y se utiliza cuando se desea obtener una medida del modelo que incorpore ambos conceptos en uno solo. Este valor se calcula con la siguiente fórmula:

$$2 * \frac{\text{Precisión} * \text{Exhaustividad}}{\text{Precisión} + \text{Exhaustividad}}$$

Usualmente, también se determina la exactitud del modelo, la cual se determina con la siguiente ecuación:

$$\frac{TP + TN}{TP + FP + TN + FN}$$

Se utilizó en esta investigación pero no es tan relevante para nosotros, ya que lo que buscamos es un modelo que sea bueno detectando únicamente barreras metálicas. por lo tanto, tenemos una predilección por conservar la mayor cantidad de imágenes que contengan barreras sin incurrir en mala clasificación. por esta razón, las métricas más importantes para nosotros son precisión y exhaustividad.

Con la exhaustividad, nos aseguramos de recuperar la mayor cantidad de imágenes con barreras posible. La precisión, por otro lado, garantiza que nuestro clasificador no devuelva demasiadas imágenes donde no hay barrera. El valor-f resume qué tan bueno es nuestro modelo en ambos aspectos.

Como podemos observar, hay varias formas de medir el desempeño de nuestro sistema. Todos estos conceptos fueron evaluados en cada una de las redes seleccionadas, sometiéndolas a la prueba de ejemplo con carreteras reales. Aunque existen muchas otras formas de evaluar clasificadores, las medidas presentadas son las más comúnmente utilizadas.

5.8 La validación

Durante la validación de nuestro modelo, emplearemos la misma estrategia utilizada en el entrenamiento de la red. Sin embargo, esta vez utilizaremos datos que la red no ha visto previamente. Mediremos su desempeño utilizando las mismas ecuaciones presentadas en la sección anterior 5.7.

El propósito de la validación es proporcionarnos una idea de cómo se comportará el modelo cuando se utilice con información nueva. En el caso del modelo diseñado en esta investigación, durante la fase de validación se le presentaron imágenes de carreteras que nunca había visto antes. Esto nos permitirá evaluar cómo se desempeña al trabajar con información nueva, una vez que ya está siendo utilizado para resolver el problema para el cual fue diseñado.

Capítulo 6

Objetivos

1. Crear bases de datos preliminares.
2. Estudiar el comportamiento de un modelo de red neuronal convolucional.
   (a) Probar diferentes valores de hiperparámetros.
   (b) Estudiar qué ocurre durante el entrenamiento y la validación.
   (c) Estudiar cómo evaluar un modelo de red neuronal convolucional.
   (d) Seleccionar una red preliminar.
      i. Realizar entrenamientos para elegir hiperparámetros.
      ii. Poner la red preliminar a prueba en condiciones ideales y de trabajo.
3. Reducir el tamaño de las bases de datos.
4. Estudiar más modelos de red neuronal convolucional.
   (a) Realizar entrenamientos para ajustar hiperparámetros.
   (b) Aplicar medidas correctivas para mejorar la precisión del modelo.
   (c) Seleccionar redes con mayor precisión y exhaustividad para elegir a la mejor.
5. Crear una interfaz con *Python* que automatice el llenado del formato de Excel utilizado para el inventario.
   (a) Incorporar la red al programa.
   (b) Implementar librerías para trabajar con Excel.
   (c) Generar un archivo único de Excel como salida de este programa.
6. Implementar "transfer learning" con el modelo más exitoso hasta el momento.
7. Seleccionar un modelo de red neuronal y recopilar resultados de clasificación.
8. Visualizar los mapas de activación del modelo.
9. Visualizar las predicciones con *Grad-CAM*.

Capítulo 7

Metodología de investigación

En este capítulo se hablará sobre cómo se alcanzaron los objetivos propuestos, comenzando por la creación de la base de datos preliminar, asumiendo una situación idealizada del problema, en la que se tiene una base de datos definida para la categoría "sin barreraz para la categoría de "barrera".

Se diseñará una Red Neuronal Convolucional que resuelva este caso con la mayor precisión posible, utilizando como punto de partida el ejemplo de Phan (2021). Esto se llevará a cabo con el propósito de comprender qué implica el entrenamiento y la evaluación de estos modelos, desde la creación de la base de datos, su limpieza y distribución, hasta el análisis estadístico de los resultados.

Durante el entrenamiento de esta primera RNC, se modificaron hiperparámetros [1] como el tamaño de lote *batch size*, el número de epochs, el número de capas ocultas, las neuronas o unidades, el *dropout* y el *spatial dropout*, con el fin de encontrar el modelo con el mejor desempeño de entre los propuestos.

A continuación, se definen los métodos para encontrar una red o RNCs que intentan dar solución al problema central de esta investigación.

Comenzando por la nomenclatura para nombrar a las redes diseñadas, se utilizó el siguiente sistema:

1. "De- *Dense*, hace referencia a la cantidad de neuronas en la capa densa.
2. "Dt- *Dropout*, esto se refiere al porcentaje asignado a este hiperparámetro.
3. "SDt- *Spatial Dropout*, esto se refiere al porcentaje asignado a este hiperparámetro.
4. Çv- *Convolutional*, esto se refiere a cuántas capas convolucionales hay.
5. "Mp- *Max pooling*, esto se refiere a cuántas capas de max pooling se tienen.

[1] Los conceptos mencionados en este párrafo no son todos definidos en este texto pero se utilizaron para mejorar el desempeño de la red.

Dimensiones y nombres de las bases de datos utilizadas:

1. S3 = Base de datos grande con 330,000 imágenes.
2. S2 = Base de datos mediana con 69, 173 imágenes.
3. S1 = Base de datos pequeña con 10,000 imágenes.

El proceso de selección preliminar implica los siguientes pasos:

1. Crear diferentes arquitecturas modificando hiperparámetros y someterlas a entrenamiento.
2. Seleccionar las redes que, en varias instancias de entrenamiento obtuvieron los valores más altos de exhaustividad, precisión, valor-f, exactitud (*accuracy*) y los valores más bajos de pérdida (*loss*).

El proceso de mejora constará de los siguientes pasos:

1. Realizar más entrenamientos con las redes seleccionadas, descartando valores de neuronas en la capa densa que no producen resultados mejores.
2. Incrementar o disminuir numéricamente el valor de hiperparámetros, como el tamaño de lote, número de capas escondidas, neuronas en la capa densa y *dropout*.

Proceso de selección final constará de los siguientes pasos:

1. Realizar varias sesiones de entrenamiento para obtener una exactitud y pérdida promedio de las redes.
2. Comparar los resultados de las redes con valores altos de exactitud y bajos de pérdida. Posteriormente, seleccionar aquella red o redes con valores más bajos de desviación estándar en relación con la exactitud promedio y la pérdida promedio.
3. Elegir la red que obtenga mejores resultados al hacer predicciones en carreteras reales despúes de estudiar el impacto de cambiar el umbral de clasificación.

Por ejemplo, comenzamos con 3 redes idénticas en términos de las neuronas en la capa densa, función de transferencia, función de activación, tamaño de lote y epochs. La diferencia estará en la cantidad de capas de convolución y *max pooling*. El primer modelo tendrá 4 pares de estas capas, el segundo 6 y el tercero 5 y una sola capa de convolución al final. De entre esos modelos se eligen dos que tengan los mejores valores de nuestras métricas a evaluar y pasamos a la siguiente fase. Ahora, de estas dos arquitecturas obtendremos modelos distintos cambiando el número de neuronas en la

capa densa y/o modificando dos hiperparámetros al mismo tiempo[2]. Luego, de este último experimento, elegimos las redes que tengan un desempeño consistente durante los entrenamientos y las ponemos a prueba clasificando imágenes de carreteras reales. Comparamos su desempeño en esta tarea y elegimos a la mejor después de estudiar su desempeño al modificar el umbral de clasificación[3].

Una vez evaluada esta red preliminar, se aumentará la cantidad de datos utilizados para entrenar a la Red Neuronal Convolucional para explorar los efectos de este cambio en los resultados. Luego, se reiniciará el proceso de selección utilizando el método preliminar de selección, mejora y selección final descrito con anterioridad.

Cuando se terminen las pruebas con la base de datos más grande, se empleará una más pequeña para explorar otros diseños de red. Este cambio también permitirá un monitoreo eficiente de los datos y su distribución. De esta manera, se ahorrará tiempo en la limpieza de la base de datos y se obtendrá una medida precisa de la clase de datos utilizados. El objetivo de este cambio es conservar las ventajas de usar más datos sin necesidad de entrenar durante tanto tiempo.

Al final, se realizarán algunas pruebas utilizando *transfer learning*. Para esto, se utilizará la red entrenada con la base de datos grande, de la cual se emplearán algunos rasgos. Luego, se agregarán algunas capas nuevas y se entrenará con una base de datos más pequeña que esté bien descrita. Posteriormente, se repetirá el proceso de selección.

[2]Se utilizó una combinación que consistía en incrementar el número de neuronas en la capa densa y agregar dropout. Esto tiene sentido, ya que al agregar neuronas, nuestro modelo tendrá una tendencia a sobreajustarse pero al incorporar dropout mitigamos este efecto.

[3]En caso de tener varios modelos que cumplan nuestras especificaciones, podemos elegir el que tenga menos parámetros. Esto se debe a que un modelo con menos parámetros resultará en un archivo más ligero, lo que también mejorará el tiempo necesario para que la red clasifique, ya que requerirá menos operaciones.

Una vez seleccionada la red neuronal para realizar las pruebas con las carreteras, se diseñará el programa para hacer el inventario de dispositivos de seguridad. Este programa utilizará únicamente las imágenes de las cámaras derecha e izquierda para que la red detecte la presencia de barreras metálicas.

El diagrama del sistema desarrollado que aplica para cualquier red es el siguiente:

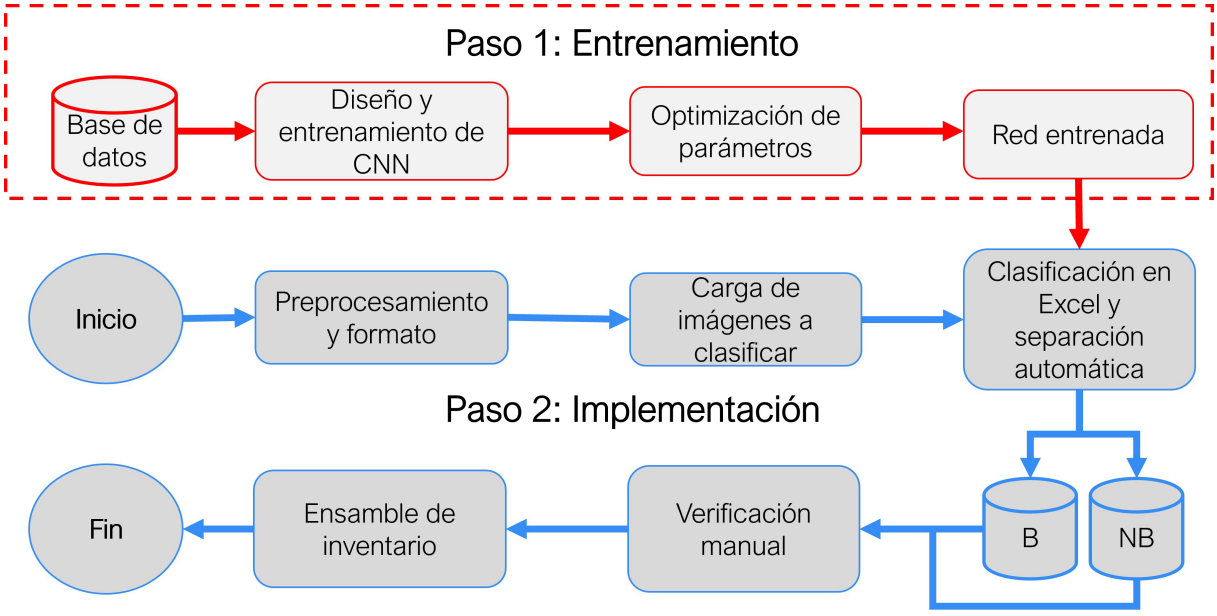


Figura 7.1: Esquema simplificado del programa elaborado

Se seleccionará al final la red que presente los valores máximos de las métricas de referencia durante las pruebas con las carreteras de ejemplo. Posteriormente, se procederá a obtener visualizaciones de sus mapas de activación. Asimismo, se obtendrán visualizaciones empleando el método *Grad-CAM*, el cual nos permite resaltar con un mapa de calor en nuestras imágenes las regiones que fueron más relevantes para el modelo a la hora de hacer la clasificación. Es decir, podemos observar en qué se fijó la red para tomar su decisión. Para obtener más información sobre este método, consulta el capitulo 8.

La prueba con 4 carreteras reales se realizará de la siguiente manera:

1. Realizar inventario a mano.
2. Identificar la posición de cada barrera en cada carretera.
3. Identificar las imágenes de barreras complicadas (aquellas con mayor obstrucción).
4. Determinar el total de imágenes a clasificar correctamente.
5. Aplicar el clasificador a la tarea.

6. Determinar el número de errores y aciertos.
7. Calcular las métricas requeridas.
8. Analizar el comportamiento del sistema al cambiar el umbral de clasificación.
9. Crear formato de salida en Excel de acuerdo al usado por la SCT.

Las carreteras que se usarán y la cantidad de imágenes por sentido para este experimento serán:

1. SEC 151-01: 1,240.
2. SEC 154-01: 1,824.
3. SEC 158-02: 1,952.
4. SEC 160-01: 944.

Capítulo 8

Desarrollo

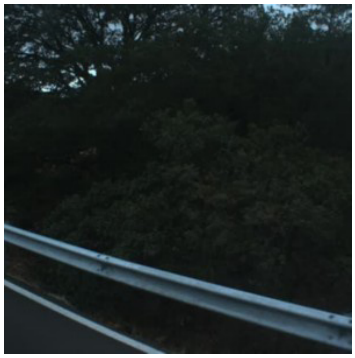
En este capítulo se detallan los pasos seguidos durante el desarrollo de la investigación para responder la cuestión planteada en la hipótesis.

Tomando en cuenta los trabajos expuestos en el capítulo de 4, se decidió únicamente usar datos de las cámaras laterales, como se hizo en la investigación de Graf, S. et al. (2019). Se tomarán los valores máximos de la investigación de Sainju, A. M. and Jiang, Z. (2020) como el objetivo a alcanzar para el clasificador propuesto, únicamente para fijar una meta, pues no se utilizan las mismas bases de datos, ni el mismo equipo. No se consideraron los resultados de Rezapour, M. and Ksaibati, K. (2021) debido a que se relacionan con categorías de barreras no contempladas en esta investigación [1]. El muestreo de las imágenes en este trabajo, así como en los mencionados, se realizó a cada 20 metros.

Se diseñará una RNC desde cero, pues en ninguno de los trabajos mencionados se usó un sistema hecho específicamente con el propósito de clasificar barreras metálicas. Esto también tendrá una finalidad didáctica para informar a miembros de la comunidad de la ingeniería civil acerca de los requisitos para crear estos sistemas.
A continuación se presentan ejemplos de la clase de imágenes que se utilizaron en la investigación. En la figura 8.1d se muestra la orilla de la carretera, que se asumió como una característica constante en las imágenes de la primera base de datos S1, para la clase de imágenes que no tenía barreras metálicas la cual se nombró NB.

La transformación que se utilizó para aumentar nuestra cantidad de datos en cada etapa de la investigación fue un efecto de espejo, similar al que se presenta en la figura 8.2. La razón de aumentar la cantidad de imágenes en las bases de datos es incrementar la variedad de rasgos a los que tiene acceso la red neuronal artificial. Esto mejora su desempeño a la hora de clasificar imágenes.

[1] Las barreras o defensas viales que usaron fueron de concreto, caja, cables e híbridas de caja con transición a cables. En esta investigación se usaron las metálicas de tres crestas, dos crestas e híbridas de tres crestas con transición a dos crestas.



(a) Dos crestas

(b) Tres crestas

(c) Dos crestas completa

(d) Orilla sin barrera

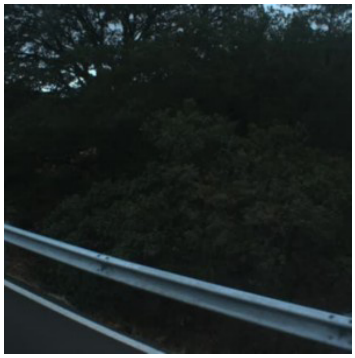
(e) Calle en ciudad

(f) Paisaje sin orilla

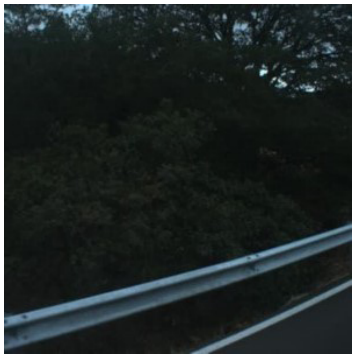
Figura 8.1: Ejemplos de imágenes usadas

8.1 Proceso con la primera base de datos S1

Para diseñar la primera red neuronal artificial que se utilizó para aprender sobre el diseño de estos sistemas y los impactos de medidas de regularización, se eligieron imágenes para crear la base de datos S1, donde siempre se presentaba una barrera o la orilla de la carretera. Esta base de datos tenía un total de 10,000 imágenes, con una distribución de 60 % para entrenamiento y el 40 % para validación, para la clase "B" (barrera) y "NB" (no barrera), como se presenta en la siguiente tabla:



(a) Imagen con barrera



(b) Imagen volteada

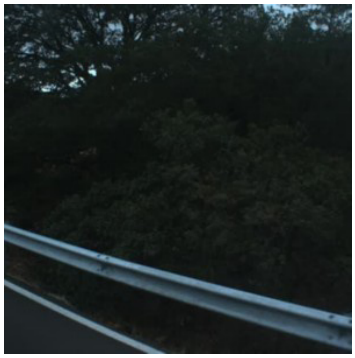
Figura 8.2: Ejemplos de transformaciones usadas

Clase	Barrera	No barrera
Etrenamiento	3000	3000
Validación	2,000	2,000
Totales	10,000	

Figura 8.3: Distribución para la base de datos preliminar S1

El primer diseño de la RNC se tomó del ejemplo presentado en Phan (2021), donde únicamente se modificaron algunos parámetros para ajustar la información a los datos recabados, y se implementó el optimizador *ADAM* mencionado en el capítulo de entrenamiento.

En la creacion de las bases de datos se usaron dos orientaciones de cámaras, lado izquierdo y derecho. Cada una de las fotografías fue tomada originalmente a cada 20 metros y luego redimensionada a 428 x 428 píxeles. Originalmente, las imágenes se presentaban en las siguientes medidas: 1624x1224 y 2048x1536 píxeles.



(a) Barrera

(b) Orilla de carretera

Figura 8.4: Ejemplos de imágenes usadas

Al final, se eligieron dos arquitecturas de Red Neuronal Convolucional de entre 20 candidatas; en la figura 8.5 se presenta una tabla con los resultados de 10 sesiones de entrenamiento.

De estas dos redes, se eligió la red con la desviación media menor. La clave para la red en la figura con la pestaña titulada "Preliminar" fue: `De568_Cv5_Mp5`. Para la red con el titulo "Red preliminar con Dropout": `De530_Dt0.3_Cv5_Mp5`. En la figura 8.5, el valor de x se refiere al valor de exactitud alcanzado por la red al final de la sesión de entrenamiento. [2]

[2]La ecuación de desviación media usada fue la siguiente $D_m = \frac{1}{N}\sum_{i=a}^{N}|x_i - \bar{x}|$ tomada de colaboradores de Wikipedia (2022)

Red preliminar		
N	**x (%)**	**$\lvert x-\bar{x}\rvert$ (%)**
1	99.4	0.433
2	99.52	0.553
3	98.87	0.097
4	99.12	0.153
5	99.15	0.183
6	99.4	0.433
7	98.08	0.887
8	97.78	1.187
9	99.27	0.303
10	99.08	0.113
Suma:	**989.67**	**4.342**
Media aritmética ($\bar{x}$):		**98.967 %**
Desviación media (D_m):		**0.4342 %**

(a) Red sin dropout

Red preliminar con Dropout		
N	**x (%)**	**$\lvert x-\bar{x}\rvert$ (%)**
1	99.23	0.034
2	99.5	0.304
3	99	0.196
4	99	0.196
5	99.33	0.134
6	99.02	0.176
7	99.33	0.134
8	99.35	0.154
9	99.1	0.096
10	99.1	0.096
Suma:	**991.96**	**1.52**
Media aritmética ($\bar{x}$):		**99.196 %**
Desviación media (D_m):		**0.152 %**

(b) Red con Dropout de 0.3

Figura 8.5: Resultados de la fase preliminar

A continuación se presenta un diagrama de las capas en la red preliminar con dropout:

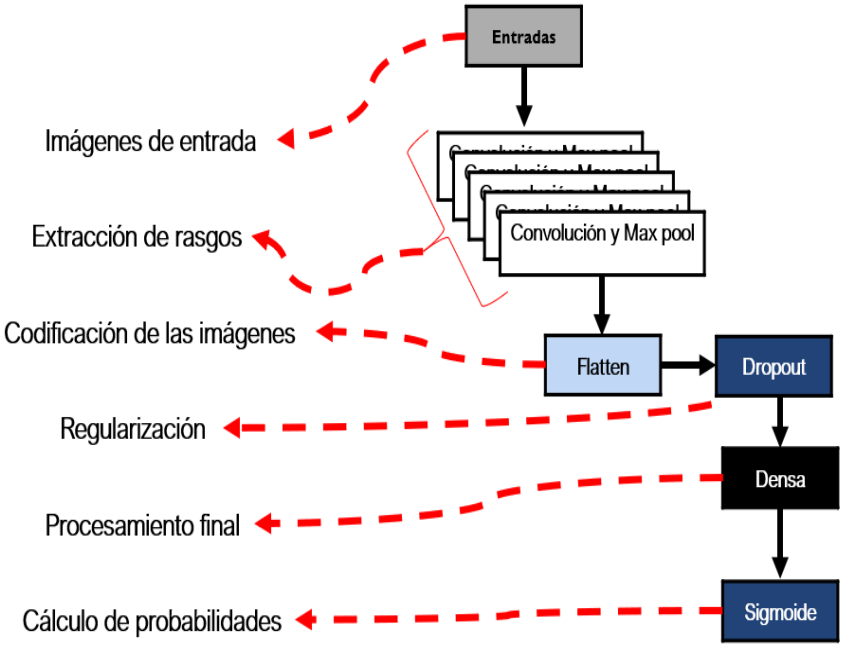


Figura 8.6: Estructura de la red preliminar con dropout

Durante esta primera prueba se añadió *dropout* y se observó una mejora considerable en los resultados de la red. Al trabajar con la base de datos S1 en esta primera parte de la investigación, aprendimos sobre el impacto de las medidas de regularización como dropout y como el agregar o remover neuronas en la capa densa afectan los resultados de nuestro modelo. El efecto de agregar o remover neuronas es el de crear un modelo simple si se remueven o uno muy complejo si se agregan demasiadas neuronas[3].

Como se explicó al principio, este proceso de determinar una buena cantidad de neuronas (una buena cantidad de neuronas es aquella que devuelve buenos resultados a la hora de evaluar el desempeño de la red con las métricas relevantes para el problema) es uno de prueba y error, donde uno puede partir tomando en cuenta la cantidad de píxeles en una de las dimensiones (alto o ancho) de las imágenes en las bases de datos.

8.2 Segunda base de datos S3

Después de estos experimentos preliminares, se decidió adaptar la red a las condiciones reales de trabajo, donde en las imágenes no tenemos una condición que siempre se cumpla, como que siempre se vea la orilla de la carretera. En esta etapa, se consideraron tres orientaciones de cámara, tanto para la clase "B" como para "NB": central, derecha e izquierda. La razón detrás de esta decisión fue que, en la actividad real,

[3]En ambos casos, este modelo tendrá desempeño poco satisfactorio medido por las métricas utilizadas en su evaluación como la exactitud, la precisión o la exhaustividad

hay ocasiones en las que una barrera no se encuentra en las imágenes de las cámaras laterales (el tamaño de las imágenes de entrada no cambió). Por lo tanto, para abordar este problema, se incluyó este ángulo extra.

(a) Cámara izquierda

(b) Cámara central

(c) Cámara derecha

Figura 8.7: Tipo de imágenes utilizadas

Es importante mencionar que el cambio de base de datos nos permite incrementar la cantidad de rasgos que nuestra red está aprendiendo de cada clase (porque tenemos imágenes nuevas de instancias donde se presenta una barrera metálica y donde no se presenta una barrera metálica), mientras más rasgos distintos aprenda nuestro modelo mejor será su desempeño identificando este objeto en imágenes que no ha visto aún. En este caso las imágenes de la base de datos S1 se incluyen en la base de datos S3 por lo cual estamos construyendo sobre los rasgos que la red ya aprendió.
Al terminar las pruebas, se obtuvieron excelentes resultados con la base de datos S3. Con esta base de datos, se entrenaron 82 modelos con rasgos diferentes.

En la siguiente tabla, podemos ver la distribución de la base de datos BDG utilizada para esta parte de la investigación:

Clase	Barrera	No barrera
Etrenamiento	99,000	99,000
Validación	66,000	66,000
Totales	330,000	

Figura 8.8: Distribución para la base de datos S3

Y la estructura de la mejor de estas redes, la cual se nombró F-mod, se presenta a continuación:

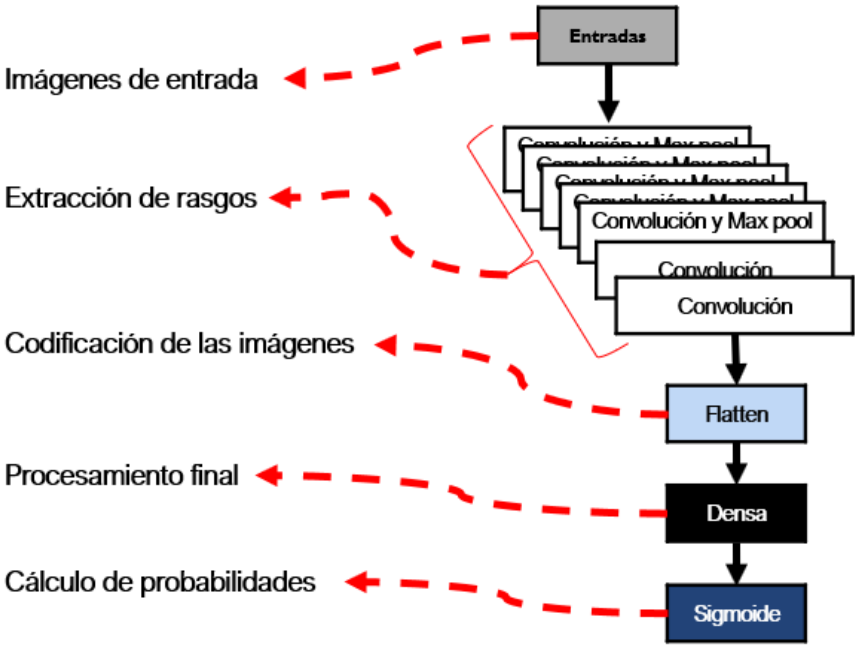


Figura 8.9: Arquitectura de F-mod

De trabajar con la base datos S3, aprendimos lo siguiente:

1. La importancia de la estructura de la base de datos para el aprendizaje del modelo: en nuestra investigación, preferimos utilizar una cantidad más alta de imágenes con barreras que de imágenes sin barreras.

2. El impacto del tamaño de la base de datos en la duración de las sesiones de entrenamiento: dado que teníamos muchas imágenes, las sesiones de entrenamiento duraban varias horas. Por lo tanto, reducir el tamaño de la base de datos se volvió muy importante para seguir adelante con la investigación.

3. La relevancia de elegir bien las métricas para evaluar el modelo pensando en el problema a resolver: en esta investigación, las métricas más relevantes son exhaustividad y precisión, lo cual no era claro desde el inicio, pero se aprendió durante el desarrollo del proyecto.

4. Utilizar métodos para la visualización de los resultados de la RNC: esto surgió a raíz de la falta de claridad en los resultados de la red, ya que realmente no sabemos qué está tomando en cuenta la red para producir el resultado que obtenemos al clasificar una imagen. Entonces, se implementaron dos métodos para lograr este objetivo: el uso de "GRAD-CAM" y la visualización de los mapas de activación en la red.

5. El efecto de la extracción de rasgos en cada capa con la operación de convolución y max pooling: observamos que al agregar capas de ambas operaciones o de una sola de ellas, el desempeño del modelo mejoraba debido al aprendizaje de patrones más pequeños presentes en las imágenes.

8.3 Base de datos final S2

Al terminar las pruebas con la base de datos S3, se decidió acortar el tiempo de entrenamiento utilizando una base de datos de menor tamaño, nombrada S2, con imágenes pequeñas (300 x 300 píxeles) utilizando únicamente las imágenes de las cámaras de la derecha y la izquierda. Esto tendría las siguientes ventajas:

1. Tendríamos una descripción clara de los datos.
2. La limpieza de la base de datos sería más rápida.
3. Podríamos experimentar con más valores de hiperparámetros.
4. Habría una probabilidad mayor de lograr visualizaciones más detalladas con *Grad-CAM*.

Entonces, se dividieron las imágenes de barrera en 4 sub categorías al hacer la base de datos. Esto se hizo con el propósito de especificar qué tipos de fotos contenían las bases de datos creadas, pero no afectó el resultado del entrenamiento. Estas categorías identificadas se presentan en la figura 8.7. La distribución que se utilizó en esta fase del proyecto fue del 70 % para entrenamiento y 30 % para validación.

Un cambio que se introdujo en esta parte del proyecto fue la idea de reducir la complejidad de los rasgos aprendidos. Esto se logró utilizando una transformación de espejo en las imágenes de la izquierda. Al hacerlas similares a las de la derecha, podríamos reducir la complejidad de los rasgos aprendidos para que nuestra red clasificara mejor (porque ahora ya no tiene que aprender la diferencia entre las fotos de la izquierda y la derecha). Esta misma técnica se aplicó a la hora de clasificar; es decir, las imágenes de la izquierda serían transformadas pero no permanentemente. Únicamente al pasar al clasificador, después de ser clasificadas, regresarían a su forma original.

(a) Barrera diagonal

(b) Extremo de barrera

(c) Barrera en esquina

(d) Barrera central

Figura 8.10: Categorías de imágenes de barreras identificadas

La distribución inicial de las imágenes en la base de datos mediana fue la siguiente:

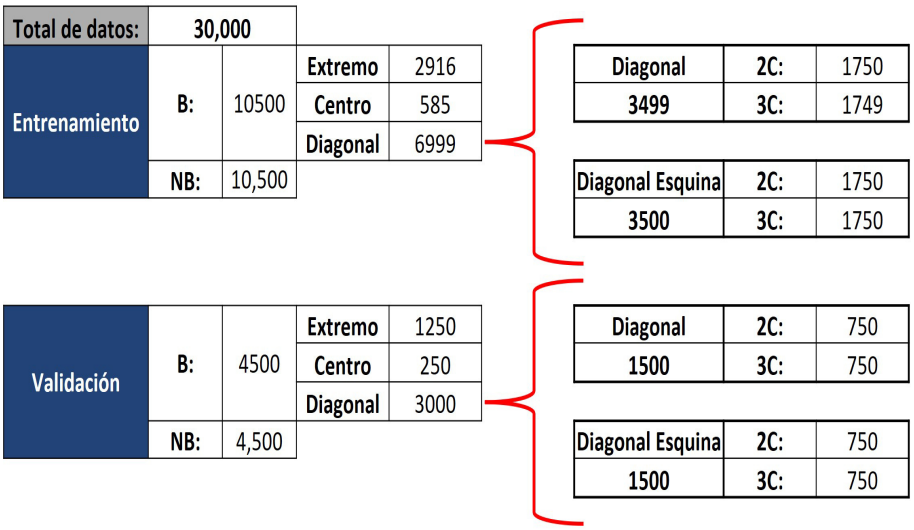


Figura 8.11: Distribución para la base de datos unilateral

Al final se retiraron algunas imágenes de la clase NB. Posteriormente, se agregaron varias imágenes de la clase B. Esto se realizó para mejorar el desempeño de la red con la clase más relevante para nosotros. Además, se modificó por primera vez el ta-

maño de los filtros empleados en las convoluciones. Esto se hizo con el propósito de mejorar la exactitud y reducir el valor de la pérdida en validación. La estructura de esta última base de datos, con la que se entrenaron 159 modelos distintos, es la siguiente:

Clase	Barrera	No barrera
Etrenamiento	29,173	19,500
Validación	12,500	8,000
Totales	69,173	

Figura 8.12: Distribución para la base de datos S2

Algo que se debe mencionar es que las imágenes de esta última base de datos se eligieron tomando en cuenta la necesidad de aprender rasgos nuevos, las imágenes donde fracasó la red F-mod a la hora de clasificar y nuestra intención de utilizar *transfer learning* para reducir la duración del entrenamiento. La ventaja de cambiar de base de datos es que puedes reducir el tiempo que duran los entrenamientos, y si utilizas *transfer learning* puedes contruir sobre rasgos que has aprendido utilizando otra base de datos sin tener que invertir la misma cantidad de tiempo o más, que fue lo que se hizo en esta investigación con el propósito de explorar nuevas arquitecturas de red.

Normalmente se utiliza una misma base de datos ya que se incluyen todas las variaciones posibles de imágenes a las que se expondrá el modelo al desempeñar su función minimizando así la pérdida de información al usar métodos como *transfer learning*. Sin embargo, esto puede resultar en entrenamientos que duren mucho tiempo, y si se desea explorar varias alternativas de arquitectura de red, esta estrategia no es viable. Entonces, si se desea explorar con nuevas arquitecturas de red lo mejor es contruir sobre los rasgos aprendidos con otras bases de datos usando metodos como transfer learning. Aunque habrá una pérdida de información, como las imágenes siguen siendo de la misma clase y poseen rasgos nuevos, se puede lograr una mejora de resultados con margen de error [4].

Expandiendo sobre la idea de usar una sola base de datos, si se tiene una buena distribucion de todos los tipos de imágenes que la conforman, evitaríamos el problema de la falta de robustez. Sin embargo, si no se posee este tipo de control y estructura, la red entrenada puede tener un sesgo fuerte hacia ciertos tipos de imagen si es que hay algún tipo que predomine. Por ejemplo, si tenemos una base de datos de perros y tenemos una cantidad predominante de una raza específica, es probable que la red no identifique bien al resto de las razas por tener un sesgo hacia este tipo de perro en específico. Otra cuestión que podría causar problemas con una sola base de datos el sobreajuste, por lo cual es beneficioso exponer la red a nuevas imágenes de la misma clase.

[4]Este error puede minimizarse al analizar los errores que comete la red para incluir imágenes donde se está equivocando al clasificar.

Como se mencionó previamente, se probaron diferentes arquitecturas de red y a continuacion tenemos los rasgos explorados en diferentes modelos:

1. Tasas diferentes para medidas de regularización en el siguiente rango 0.15-0.5.
2. Más o menos operaciones de max-pool y convolución en el rango siguiente: 2 a 3 pares o capas de una sola operación.
3. Más o menos neuronas en la capa densa en el siguiente rango: 400-560 neuronas en la capa, agregando o restando neuronas en incrementos de 10 neuronas según el desempeño del modelo.
4. Funciones distintas de transferencia como ReLU, sigmoide, tanh (tangente hiperbólica).
5. Diferentes tamaños de filtros convolucionales en un rango de: 2x2 - 4x4.
6. Diferentes cantidades de filtros convolucionales en un rango de 8-256.
7. Más o menos epochs en un rango de: 10-20.

8.4 Red C-mod

Antes de elegir una red definitiva, se decidió aplicar *transfer learning* usando como modelo base la mejor de las redes neuronales hasta el momento, F-mod (La cual se entrenó con la base de datos grande).
Para diseñar una nueva red que hiciera uso de los rasgos aprendidos por esta primera, la red que utilizó rasgos de F-mod se nombró C-mod y su arquitectura se presenta a continuación en la figura 8.13:

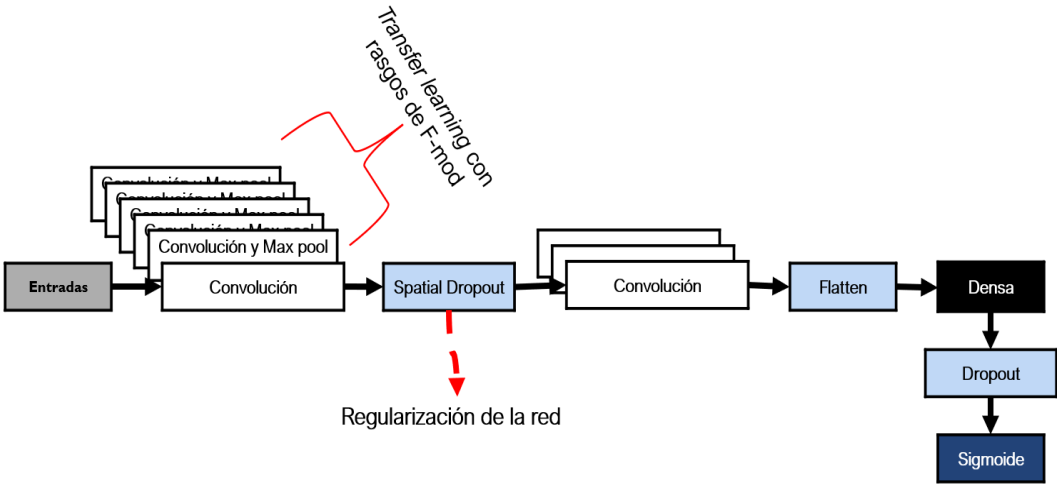


Figura 8.13: Arquitectura de C-mod

Los resultados obtenidos en el entrenamiento de la red C-mod para la métrica de exactitud se registraron en la siguiente tabla:

C-mod con Transfer Learning		
N	**x (%)**	**$\lvert x-\bar{x}\rvert$ (%)**
1	99.4	0.433
2	99.52	0.553
3	98.87	0.097
4	99.12	0.153
5	99.15	0.183
6	99.4	0.433
7	98.08	0.887
8	97.78	1.187
9	99.27	0.303
10	99.08	0.113
Suma:	**989.67**	**4.342**
Media aritmética ($\bar{x}$):		**98.967 %**
Desviación media (D_m):		**0.4342 %**

Figura 8.14: Exactitud promedio para red C-mod

Para usar "*transfer learning*", se utilizó la información presentada en Mitiku. (2014) para crear el código. De acuerdo con Marcelino (2018), la idea de aplicar este método,

preservando algunas capas, entrenando otras y agregando unas nuevas, se considera práctica, pues se tienen relativamente pocos parámetros y una base de datos grande, por lo cual el sobreajuste no es un problema.

Siendo especificos, se tomó el modelo F-mod. Se mantuvieron los rasgos de las primeras 8 capas, se eliminaron las últimas 5 capas y se añadieron 8 capas nuevas con 535 neuronas en la capa densa. Además, se agregó "*spatial dropout*" y "*dropout*".

Antes de mencionar que red se eligió, debemos hablar sobre el umbral de decisión, que es el valor que nos devuelve la red como resultado de la clasificación. Este valor representa la probabilidad de que la imagen clasificada pertenezca a una categoría o a otra, y podemos elegirlo en base a nuestro criterio de selección; por ejemplo, en esta investigación utilizamos 70 % o 0.7 para determinar si una imagen contiene una barrera. Por lo tanto, si tenemos una imagen con un valor de salida de 0.6 o 60 %, se clasificará como perteneciente a la categoría NB (No barrera), lo que indica que no hay una barrera en la imagen procesada.

La red propuesta como la mejor surgió como resultado de comparar el desempeño de la red `De535_SDt0.4_Dt0.15_Cv9_Mp5` (C-mod) y `De530_nDt_Cv7_Mp5` (F-mod) al resolver la actividad de ejemplo, que consistió en inventariar 4 carreteras secundarias. Para ambas redes, se utilizó un umbral de decisión del 70 %. Los resultados de la prueba final se presentan a continuación:

Resultados con umbral (U%) de 50%		
Métricas	**F-mod**	**C-mod**
Precisión	95.69%	58.09%
Exhaustividad	74.22%	**94.63%**
Valor-F	82.01%	67.33%
Barreras detectadas de 232	197	**219**
Falsos positivos	8	**150**

Figura 8.15: Ultimos entrenamientos

A continuación tenemos la representación gráfica de los valores en la tabla anterior[5].

Las gráficas 8.16 y 8.17 muestran los resultados de la red F-mod, la cual se destaca por su capacidad para clasificar con un umbral de decisión alto sin reportar falsos positivos (al menos con el conjunto de imágenes de la actividad de ejemplo). Esto podría permitir omitir el paso de revisión manual, lo que reduciría aún más la duración del inventariado.

En las siguientes gráficas, la curva en color negro fuera del recuadro representa la proporción de valores que se encuentran en la dimensión horizontal del gráfico.

[5]Para las gráficas se utilizó solo un 10 % de los verdaderos negativos.

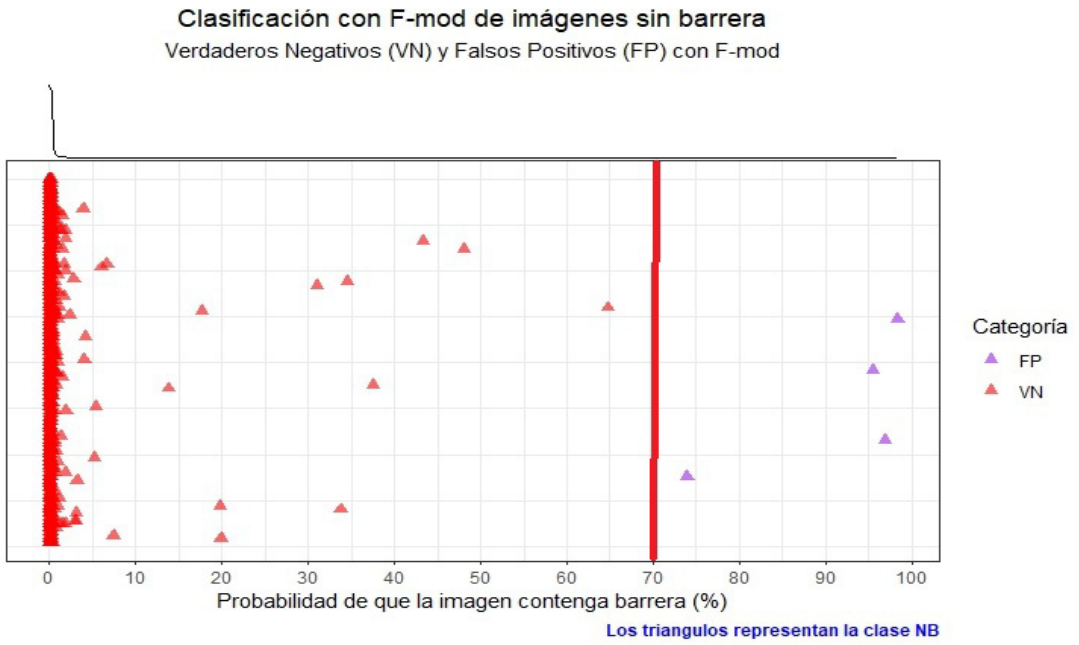


Figura 8.16: Resultados para imágenes sin barrera

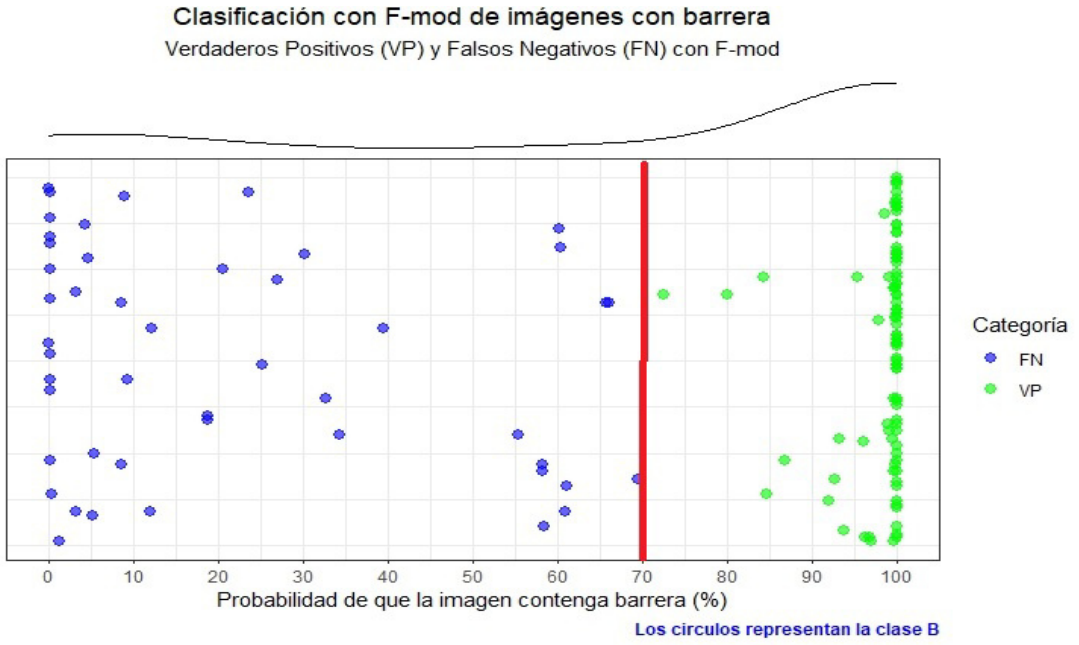


Figura 8.17: Resultados para imágenes con barrera

Las gráficas 8.18 y 8.19 muestran los resultados obtenidos con la red C-mod en la actividad de ejemplo. Una característica de esta red es que devuelve una proporción más alta de imágenes con barrera en comparación con la red F-mod. Sin embargo, esto va acompañado de algunos errores, aunque la cantidad de imágenes sin barrera correctamente clasificadas es pequeña en comparación. Esto hace que el uso de esta red sea conveniente.

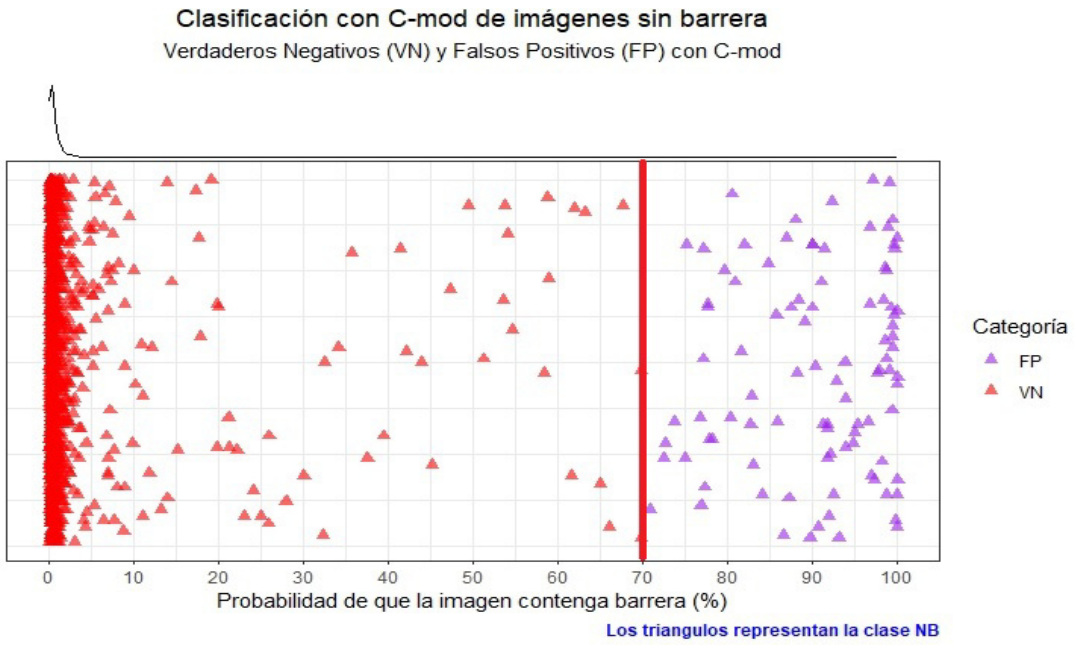


Figura 8.18: Resultados para imágenes sin barrera

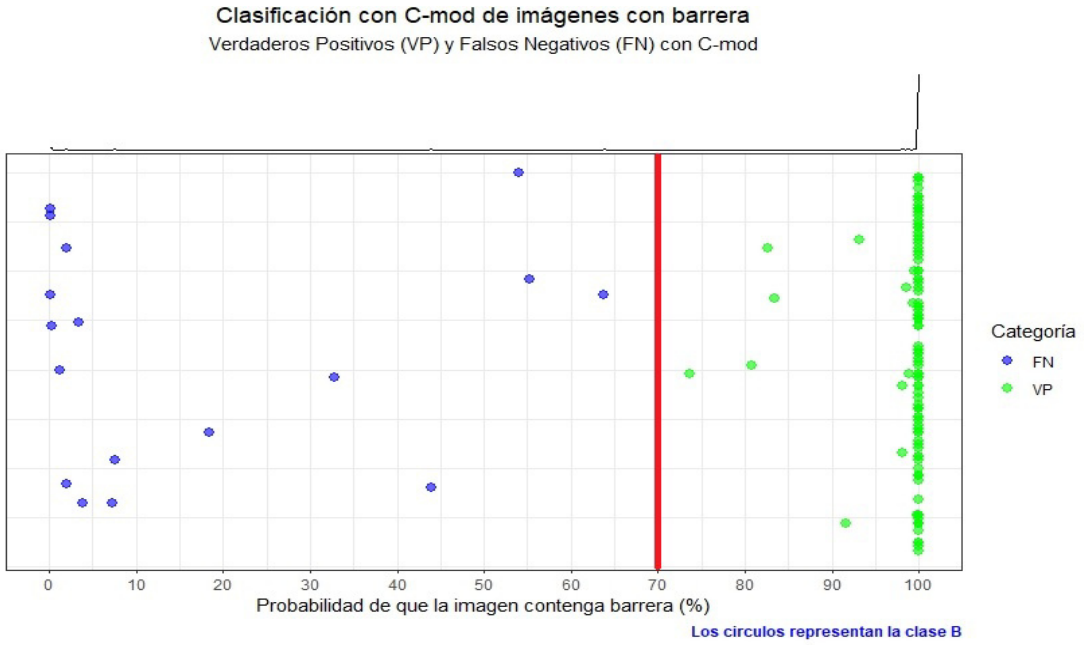


Figura 8.19: Resultados para imágenes con barrera

Como pudimos observar en las gráficas 8.18 y 8.19, la red C-mod nos devuelve una mayor cantidad de imágenes de barrera correctamente clasificadas, aunque a costa de varios "falsos positivos". Sin embargo, en perspectiva, son pocas las imágenes incorrectamente clasificadas. Por otro lado, la red F-mod detecta pocos "falsos positivos", pero nos devuelve una cantidad menor de imágenes correctamente clasificadas. En el contexto del inventariado, es preferible un sistema que identifique más imágenes con barrera, aunque pueda cometer algunos errores.

Al final, se eligió la red C-mod. En un análisis posterior, mencionado en el siguiente capítulo, se refuerza esta decisión con una condición importante al momento de llevar el modelo a la aplicación práctica: modificar el valor del umbral de decisión [6].

La figura 8.20 muestra una comparación de los resultados de la prueba con las carreteras reales, con el fin de ilustrar el rendimiento del sistema diseñado. La comparación se hizo entre los valores máximos de Sainju, A. M. and Jiang, Z. (2020) utilizados como referencia, y la red C-mod con un umbral de 70 %.

[9] Sainju, A. M. and Jiang, Z. (2020)		Valores obtenidos por la red C-mod usando U (%) = 70	Diferencia
Valor-F	**0.84**	0.74	-0.10
Exhaustividad	**0.78**	0.93	0.15
Precisión	**0.91**	0.66	-0.25

Figura 8.20: Comparativa con referencia

[6]Sabemos que la red nos devolverá una probabilidad. Con el umbral de decisión, establecemos un criterio de aceptación teniendo en cuenta esta probabilidad. Por ejemplo, si nuestro umbral (U %) es del 50 % el criterio de clasificación podría ser: "Si la probabilidad de salida es <= 50 %, esta imagen pertenece a la clase NB; de lo contrario, es de la clase B"

Para comprender mejor el proceso interno del sistema diseñado, se realizaron las siguientes visualizaciones de los filtros aprendidos y los mapas de activación con la red `De535_SDt0.4_Cv9_Mp5`. Para crear estas imágenes de los filtros aprendidos por la red[7] y los mapas de activación, se utilizó el ejemplo presentado por Khandelwal (2020), del cual únicamente se modificaron algunas lineas del código para adaptarlo a los datos utilizados.

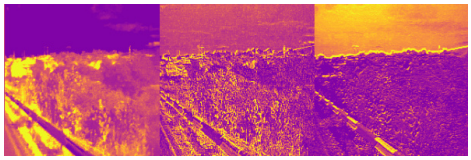
Figura 8.21: Ejemplos de filtros aprendidos por la red

A continuación se presentan dos ejemplos[8] de mapas de activación producidos por los filtros aprendidos por la red. Las imágenes de referencia se muestran en 8.24a para la clase B y en 8.25a para la clase NB.

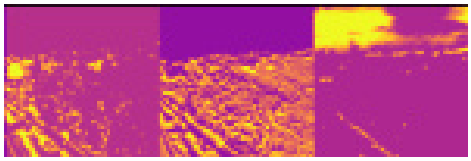
En esta parte de la investigación se mejoró el diseño del sistema al utilizar únicamente imágenes de una orientación de cámara, esto se hizo para que la red no aprendiera otro patrón al usar dos orientaciones reduciendo la complejidad de la tarea de aprendizaje, se exploraron muchas arquitecturas nuevas gracias a que los entrenamientos duraban menos, se trabajó con métodos para crear visualizaciones para entender a detalle las

[7] En el ejemplo se presentan los primeros 16 filtros de 3x3 aprendidos por la RNC, se utiliza el mapeo "Viridis" de development team (2022) para colorear la imagen, donde valores brillantes corresponden a valores de peso w_i alto y los tonos oscuros a valores bajos.

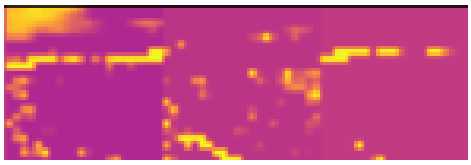
[8] En estos ejemplos se utilizó el mapeo "Plasma" presentado en development team (2022), en donde los tonos brillantes corresponden a valores altos, y los oscuros a valores pequeños de activación.



(a) Primera convolución



(b) Max-pooling 2

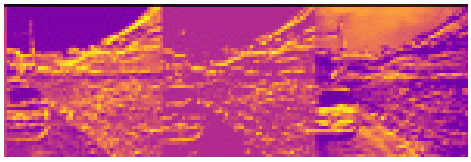


(c) Max-pooling 4

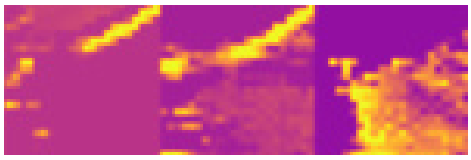
(d) Última convolución

Figura 8.22: Mapas de activación para una imagen con barrera

(a) Primera convolución



(b) Max-pooling 2



(c) Max-pooling 4

(d) Última convolución

Figura 8.23: Mapas de activación para una imagen sin barrera

partes del sistema diseñado, y se mejoró sobre los resultados obtenidos durante la primera parte de este proyecto de investigación.

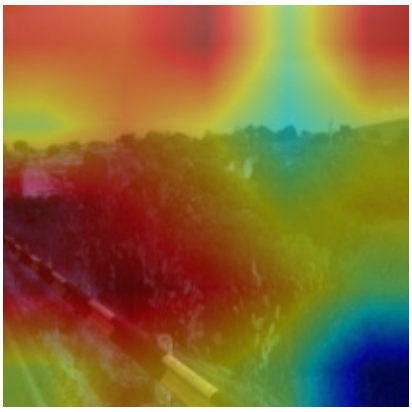
8.5 GRAD-CAM

Pero, ¿Qué es *Grad-CAM* ?, como lo presentan los autores en Ramprasaath, S. et al. (2019) "*A technique for producing "visual explanations" for decisions*", es decir una forma de explicar visualmente la decisión de una RNC.

Para su implementación, se siguió el ejemplo presentado por Innat (2021).

De acuerdo con Draelos (2022), el proceso que realiza *Grad-CAM* de forma simplificada es obtener un mapa de calor[9] que muestra la región de mayor importancia para la decisión de la red, [10] como podemos observar en la figura 8.24.

(a) Imagen de entrada



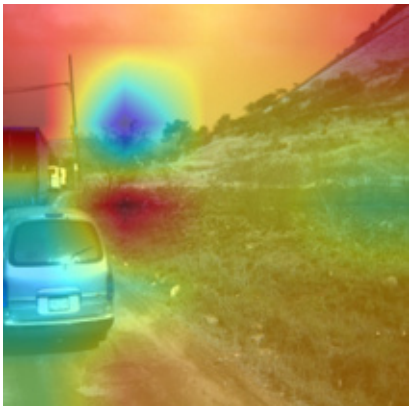
(b) Región relevante para la decisión

Figura 8.24: Aplicación de *Grad-CAM* en imagen de la clase B

[9]Para esta visualización, se utilizó el mapeo "JET" de development team (2022). En el mapa de calor de los ejemplos, los valores en rojo representan mayor relevancia, mientras que los azules menor relevancia.

[10]Esta región de importancia se obtiene a través de una suma ponderada, donde se incrementa el valor de los mapas de activación que influyeron más en el valor que se convierte en la probabilidad de clase, que se obtiene de la última función de activación en la red.

(a) Imagen de entrada



(b) Región relevante para la decisión

Figura 8.25: Aplicación de *Grad-CAM* en imagen de la clase NB

Desafortunadamente, en esta investigación GRAD-CAM no resultó útil debido a que no se toma en cuenta únicamente el objeto de interés para realizar la clasificación. En la figura 8.26 podemos ver un ejemplo presentado en Keras (2020) de como luce una imagen en donde la herramienta funciona adecuadamente:

(a) Imagen de entrada

(b) Región relevante para la decisión

Figura 8.26: Aplicación de *Grad-CAM* en imagen de perro y gato

8.6 Hardware

En esta investigación se compararon los tiempos de entrenamiento de una red entre un equipo que poseía *GPUs (Graphic Processing Units)* y otro que no tenía estas unidades. Todo esto fue posible gracias al apoyo del Laboratorio de Supercómputo del Bajío ya que nos dieron acceso a estos equipos para realizar esta investigación.

las características de los equipos que nos facilitaron son las siguientes:

1. n-gpu01: Tarjeta gráfica Nvidia k40 con 12 nodos de GPU, 2 x Procesadores Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz y 64GB Memoria RAM.
2. n-quad: 4 x Procesadores Intel(R) Xeon(R) CPU E5-4627 v4 @ 2.60GHz y 1 TB de Memoria RAM.

Los resultados de una sesión de entrenamiento de la misma red en ambos equipos se presenta en la tabla de la figura 8.27.

Máquinas:	n-gpu01 = a	n-quad = b	a/b
# Epoch	Horas	Horas	
1	2.8	3.5	0.8
2	2.2	3.5	0.6
3	2.1	3.5	0.6
4	2.1	3.5	0.6
5	2.1	3.5	0.6
6	2.1	3.4	0.6
7	2.1	3.4	0.6
8	2.1	3.4	0.6
Duración de entrenamiento:	17.8	28.1	0.6
Promedio de tiempo/epoch:	2.2	3.5	0.6

(a) Equipos y tiempos

Horas ahorradas	10.3
Horas/epoch:	1.3

(b) Ahorros totales

Figura 8.27: Comparación entre dos equipos a la hora de entrenar.

Solo la red preliminar y algunos entrenamientos de la red F-mod se realizaron con el equipo que no tenía GPUs. El resto de los entrenamientos de F-mod y los de modelos subsecuentes, como C-mod, se realizaron con el equipo n-gpu01 que posee GPUs. Esto redujo la cantidad de tiempo invertido en las sesiones de entrenamiento y también nos permitió explorar diferentes ideas relacionadas con la arquitectura de la red.

Capítulo 9

Análisis y discusión

El resultado de inventariar 4 carreteras usando la red F-mod desarrollada en este trabajo fue una reducción en el tiempo de ejecución del 90.21 %. Este valor surge de comparar el tiempo necesario para realizar la tarea manualmente (3 horas con 14 minutos), con la RNC y las librerías auxiliares desarrolladas para leer y registrar los valores (19 minutos). El cálculo del porcentaje se hizo siguiendo el procedimiento de The BBC (2022).

Enseguida se presentan los detalles; usando las variables:

1. Tiempo a mano: T_m
2. Tiempo con programa: T_p
3. Diferencia: D
4. Reducción: R
5. Porcentaje de Reducción: $\%R$

El procedimiento fue el siguiente:

1. Convertimos las horas a minutos: $T_m = 3\,horas * \dfrac{60\,minutos}{1\,horas} = 180\,minutos$
2. Determinamos el nuevo tiempo a mano: $T_m = 180\,minutos + 14\,minutos = 194\,minutos$
3. Calculamos diferencia: $D = T_m - T_p = 194\,minutos - 19\,minutos = 175\,minutos$
4. Calculamos la reducción: $R = \dfrac{D}{T_m} = \dfrac{175\,minutos}{194\,minutos} = 0{,}90206$
5. Obtenemos el porcentaje: $\%R = 0{,}90206 * 100\,\% = 90{,}21\,\%$

El estudio de la red continuó más allá de los resultados de los entrenamientos y la prueba con datos de carreteras reales. Al notar que la red aún cometía algunos errores al clasificar imágenes de la categoría de interés (B), se estudió el impacto del umbral

de decisión, es decir, el porcentaje de probabilidad que separa las clases. Esto significa que ciertos resultados de probabilidad a la hora de clasificar una imagen serán aceptados o serán rechazados con base en este umbral.

En otras palabras, podemos volver más estricta la decisión de que valores de probabilidad aceptaremos para la clase de interés (B). Por lo tanto, lo que se realizó fue ver el efecto de cambiar el umbral (U %) en las dos redes finales. Así fue como se eligió la red "C-mod", ya que sin necesidad de incrementar tanto el umbral, logramos obtener más imágenes de barreras sin cometer tantos errores. Sin embargo, el clasificador más preciso es "F-mod", solo que nos devuelve menos información relevante. Al final, el objetivo es registrar correctamente la mayor cantidad de imágenes con barreras metálicas posibles.

En el anexo D podemos encontrar las gráficas del experimento con distintos valores de umbral para la categoría "B", para las redes que se mencionaron en el párrafo anterior.

Por el momento, el resultado del programa desarrollado para replicar el formato de la dependencia únicamente contempla algunos rubros sobre los cuales podíamos obtener la información. Se planea expandir el alcance del trabajo posteriormente para poder incluir otros datos, como los valores de altitud y latitud.

En el anexo F podemos encontrar el formato generado de forma automática por el programa creado, junto con una codificación de colores que contempla los diferentes rubros que integran el inventario, aquellos que actualmente el programa llena automáticamente y los que no.

Los resultados expuestos por el momento se consideran buenos debido al alcance de este proyecto. Se aprendieron algunas cuestiones durante la creación de la RNC empleada, especialmente porque la solución se diseñó para una clase de imágenes muy específica. Al principio, hubo algunas consideraciones desconocidas que se implementaron más tarde en la investigación, esto sucedió debido a que se estaba aprendiendo a cómo trabajar con estas estructuras.

Para ser específicos, las consideraciones mencionadas se relacionan con el tamaño de las imágenes usadas, el tamaño de los filtros convolucionales, el tamaño del paso de estos al procesar los datos, la distribución de las bases de datos, la clase de modificaciones que se pueden usar para incrementar la cantidad de rasgos a extraer durante el entrenamiento y, por último, la calidad de los datos usados.

Se considera que al usar imágenes un poco más grandes [1], de mejor calidad, con transformaciones relacionadas con los tonos de la imagen, con una distribución sesgada para la clase de interés y con un estudio más riguroso de los efectos del tamaño

[1]Se usó un tamaño de 428x428. Entonces podemos intentar con 640x640 recortando sin alterar calidad al comprimir la imagen, se usaría este valor pues puede ser dividido en dos varias veces, y podemos preservar la dimensión usando relleno (el relleno o *zero-padding* es agregar ceros en los bordes de la imagen).

de los filtros y el valor del paso cuando estos procesen los datos, se podría diseñar un sistema que sea más exitoso. Estos son los rubros donde se encontró potencial, pero también algunas complicaciones. Claro que el efecto de estas modificaciones se estudiará en una investigación posterior.

Una opción que se puede implementar en un trabajo futuro es usar side-tuning, el cual, según Jeffrey, O. et al. (2020), consiste en emplear un clasificador ya entrenado junto con una red alternativa más pequeña. Esta última se fusiona con la red ya entrenada mediante una suma. Según describen los autores, este método es menos propenso a sobreajustarse y no sufre de la catástrofe de olvidar al realizar aprendizaje incremental.

Otra opción es utilizar una red preentrenada y aplicar *transfer learning* en lugar de diseñar una red desde cero. Claro, también podríamos iniciar el proceso de diseño de la red desde cero, teniendo en cuenta todo lo aprendido en este proyecto y revisando aspectos que no se exploraron a profundidad en este trabajo.

Capítulo 10

Conclusiones

Basándonos en lo expuesto durante el desarrollo, podemos afirmar que se ha logrado demostrar veracidad de la hipótesis de investigación planteada al inicio. Esta hipótesis se centraba únicamente en determinar si era posible automatizar el inventariado de barreras metálicas. Sin embargo, hemos descubierto un potencial aún mayor al observar la velocidad de la solución en comparación con el método actual. Podría decirse que esto es solo el comienzo, ya que hay margen para ir más allá. No obstante, por el momento, los resultados son satisfactorios.

Es importante tener en cuenta que se ha utilizado un diseño sencillo de RNC en comparación con el estado del arte. Los sistemas modernos utilizan estructuras muy diferentes a las propuestas en esta investigación, por lo tanto, aún existe la posibilidad de mejorar considerablemente los resultados expuestos en este trabajo.

Teniendo en cuenta las lecciones aprendidas durante el proyecto, no tengo dudas de que un sistema como este puede lograr automatizar casi por completo la actividad del inventariado de dispositivos de seguridad de la SCT. A partir del desarrollo de este sistema, queda claro que este método también puede aplicarse a otras actividades dentro de la misma dependencia, como el inventariado de señales de tráfico.

Por último, espero que este trabajo inspire a otros estudiantes a considerar la relación entre la ingeniería civil y otras áreas de estudio. Esto con el propósito de ampliar el repertorio de los ingenieros para resolver problemas que, hasta el momento, no se han abordado desde el ángulo correcto.

Apéndice A

Propagación hacia atrás

En este apartado se presenta la propagación hacia atrás o diferenciación en reversa, también conocida en inglés como "*backpropagation*".

La figura A.1 representa la ecuación $e = c * d$ para hallar la variable e. El proceso de descomponerla en operaciones y crear una gráfica para analizarla. Se consideró trivial la evaluación en cada nodo de la expresión, por lo cual se pasó directamente a los gradientes:

Según Olah (2014a), la regla para realizar derivadas en la gráfica es sumar las ramas conectadas y multiplicar los nodos conectados en las ramas.

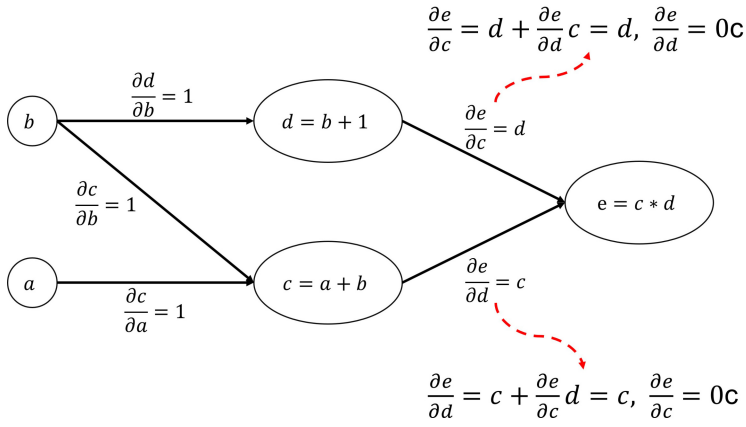
$e = (a + b) * (b + 1)$

c= $(a + b)$

d= $(b + 1)$

$e = c * d$

(a) Definiciones



(b) Gráfica

Figura A.1: Gradientes de las variables conectadas

Entonces, si quisiéramos obtener, por ejemplo, la derivada de e respecto a b, el procedimiento hacia adelante o de derivación convencional sería el presentado en la figura A.2 y se utilizaría el diagrama de la figura A.3, donde se obtiene la derivada $\frac{\partial e}{\partial b}$:

$$\frac{\partial e}{\partial b} = \frac{\partial d}{\partial b} * \frac{\partial e}{\partial c} + \frac{\partial c}{\partial b} * \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial b} = 1 * \frac{\partial e}{\partial c} + 1 \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial b} = d + c$$

Figura A.2: Aplicación de la regla

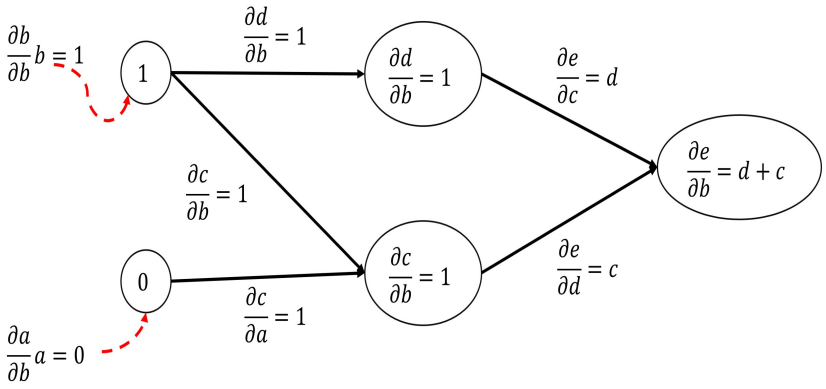


Figura A.3: Gráfica resultante

Tomando en cuenta el resultado y la regla del proceso, nos damos cuenta de que aquí se manifiesta la regla de la cadena.

La propagación hacia atrás se presenta a continuación en la figura A.4:

$$\frac{\partial e}{\partial b} = \frac{\partial d}{\partial b} * \frac{\partial e}{\partial c} + \frac{\partial c}{\partial b} * \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial b} = 1 * \frac{\partial e}{\partial c} + 1 * \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial b} = c + d$$

(a) Cambios en e por b

$$\frac{\partial e}{\partial a} = \frac{\partial d}{\partial a} * \frac{\partial e}{\partial c} + \frac{\partial c}{\partial a} * \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial a} = 0 * \frac{\partial e}{\partial c} + 1 * \frac{\partial e}{\partial d}$$

$$\frac{\partial e}{\partial a} = 1 * \frac{\partial e}{\partial d} = c$$

(b) Cambios en e por a

Figura A.4: Derivadas de las entradas

Como podemos ver en la figura A.5, el resultado de aplicar la propagación hacia atrás son todas las derivadas de la salida en relación a todos los nodos de la gráfica. Esto

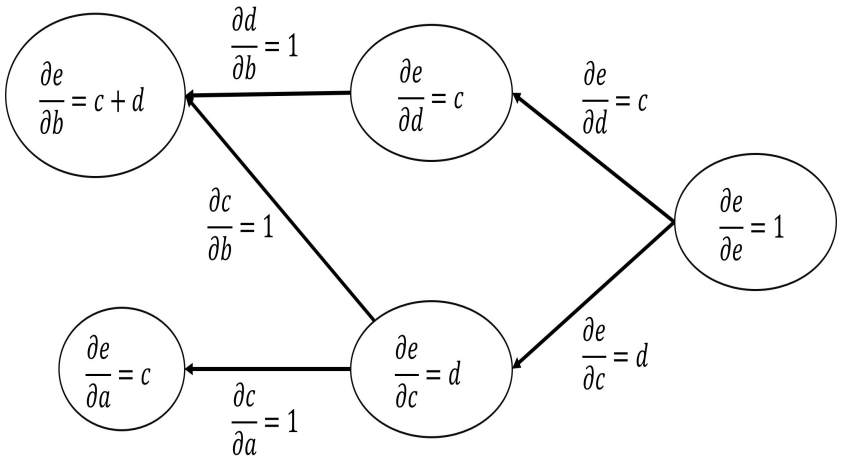


Figura A.5: Gráfica resultante

es útil en el entrenamiento de sistemas como el desarrollado, ya que ahorra mucho tiempo de cálculo conforme crece la complejidad de la gráfica.

En este caso, podemos pensar en la variable e como la probabilidad de clase asociada a una imagen de entrada, y las derivadas que obtuvimos como la contribución del error de cada peso correspondiente a las neuronas que produjeron esa probabilidad al activarse. De este modo, sabemos cómo cambiar estos pesos a través del optimizador para acercarnos a la clasificación correcta de la imagen de entrada.

Apéndice B

Gradiente de error

A continuación, en la figura B.1 se presenta un ejemplo de cómo obtener el gradiente de error de la función de costo ECM (Error Cuadrático Medio), utilizando el optimizador descenso de gradiente.

Las flechas indican la dirección de lectura y los números indican las fases en las que se dividió el proceso.

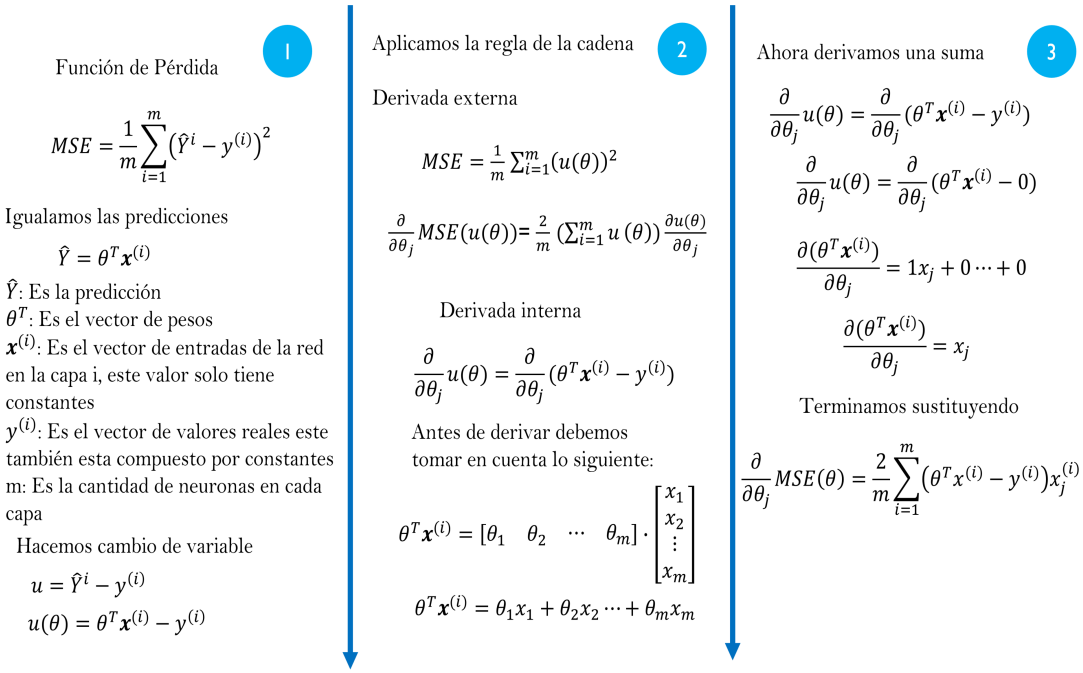


Figura B.1: Procedimiento de optimización

Apéndice C

Red totalmente conectada

En este apartado se hablará sobre como una red neuronal totalmente conectada, como la que se muestra en la figura C.1, puede clasificar entre dos clases. Olah (2014b) presenta el caso de una red neuronal simple que clasifica los puntos de dos curvas. La red está compuesta por una capa de entradas, una capa oculta y una capa de salida. La cantidad de neuronas en cada capa sí importa, pero nos centraremos únicamente en el proceso que realizan estos sistemas.

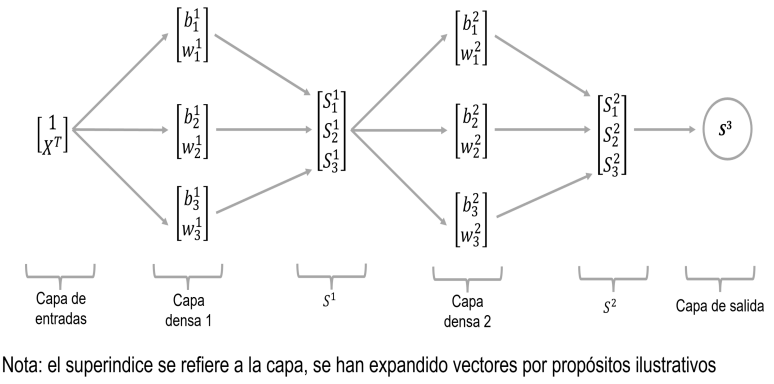


Figura C.1: Esquema de red neuronal totalmente conectada

Antes de hablar sobre cómo se logran separar los datos, debemos presentar la ecuación más importante durante la explicación, pues nos permite determinar la frontera de decisión. Esta se muestra en la siguiente figura:

$$h_{W,b}(X) = \varphi(X^T W + b)$$

W: Matriz de pesos

X: Matriz de entradas

b: Vector de sesgos (*bias*)

φ: Función de activación

$h_{W,b}$: Salida de las conexiones

Figura C.2: Ecuación principal

Entonces, como pudimos ver en la imagen anterior, la ecuación se asemeja mucho a la de una recta siedo afectada por una función, el efecto que esto tiene en los datos será explicado a continuación con un ejemplo sencillo. Supongamos que tenemos dos clases (por ejemplo, dos frutas como uvas y moras). Trabajaremos en dos dimensiones, pues estaremos midiendo dos tipos de rasgos (por ejemplo, altura y peso), por lo tanto, necesitaremos dos ejes para cuantificar ambas características. A continuación, en la figura C.3, se muestra la simbología que se usará para ambas clases.

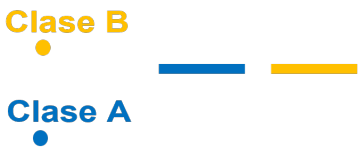
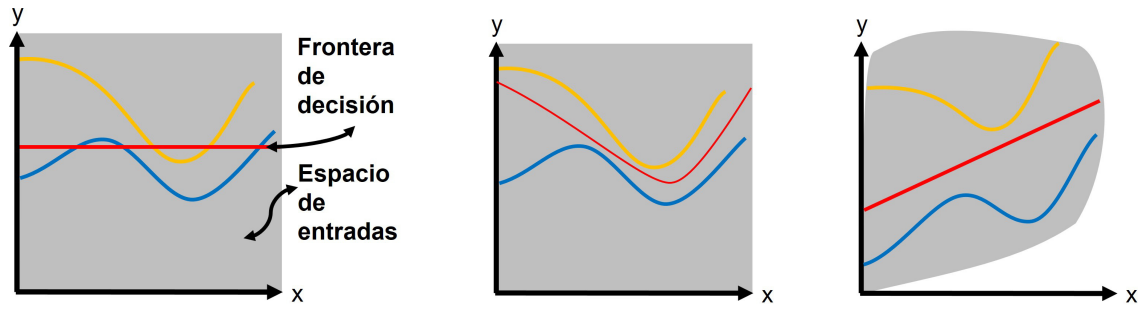


Figura C.3: Datos a clasificar

El ejemplo consiste en determinar qué puntos pertenecen a cuál curva (es decir, cuando tenemos uvas y cuándo moras). Una red sin capas ocultas haría la aproximación mostrada en C.4a, una red con una capa escondida crearía una frontera de decisión más compleja como se muestra en la figura C.4b. La imagen anterior es una representación previa a la aplicación de la función de activación. Al aplicar esta función a los datos de entrada en la capa oculta, obtendremos lo presentado en la figura C.4c. Como podemos observar con esta última representación, es más fácil trazar una recta que separe los datos.

Como lo menciona Olah (2014b), con cada capa, la red transforma los datos creando una nueva representación [1] de los mismos. Entonces, cuando lleguemos a la última

[1] En la siguiente página se muestran visualizaciones de estas transformaciones en los datos: https://cs.stanford.edu/people/karpathy/convnetjs//demo/classify2d.html



(a) . (b) . (c) .

Figura C.4: Representaciones internas de los datos

capa, esta traza una línea (o en dimensiones superiores un hiperplano) que separa los datos.
Es importante comprender cómo las operaciones afectan al espacio de entradas, por lo tanto, se enumeran a continuación:

1. Transformación lineal dada por el producto con la matriz W.
2. Traslación dada por el vector b.
3. Aplicación puntual de la función de activación, Olah (2014b) presenta la función tanh (tangente hiperbólica).

Las transformaciones que se realizan en cada capa estiran y contraen el espacio, pero nunca lo cortan, rompen o doblan, preservando así sus propiedades topológicas. [2] [3]

Olah (2014b) menciona que la clasificación con una unidad sigmoide o usando una capa de softmax es equivalente a tratar de encontrar un hiperplano[4] (o en este, caso una línea) que separe las clases A y B en la representación final.

Ahora visualicemos el proceso en tres dimensiones (por ejemplo, mediremos tres características de las dos cosas que queremos clasificar, por lo tanto, requerimos tres ejes). La simbología se presenta en la parte superior derecha de las gráficas. Utilizaremos como base el ejemplo presentado por Olah (2014b), y el resultado se presenta en la siguiente secuencia de figuras:

[2] De acuerdo con colaboradores de Wikipedia (2021), “una propiedad topológica es una propiedad de un espacio topológico que es invariante bajo un homeomorfismo".

[3] Según colaboradores de Wikipedia (2020), una definición informal de homeomorfismo es que refleja cómo dos espacios topológicos son "los mismos"vistos de otra manera. El ejemplo mencionado en la referencia es una taza y una dona.

[4] colaboradores de Wikipedia (2019) presentan esta explicación del concepto: “En un espacio unidimensional (como una recta), un hiperplano es un punto: divide una línea en dos líneas. En un espacio bidimensional (como el plano xy), un hiperplano es una recta: divide el plano en dos mitades."

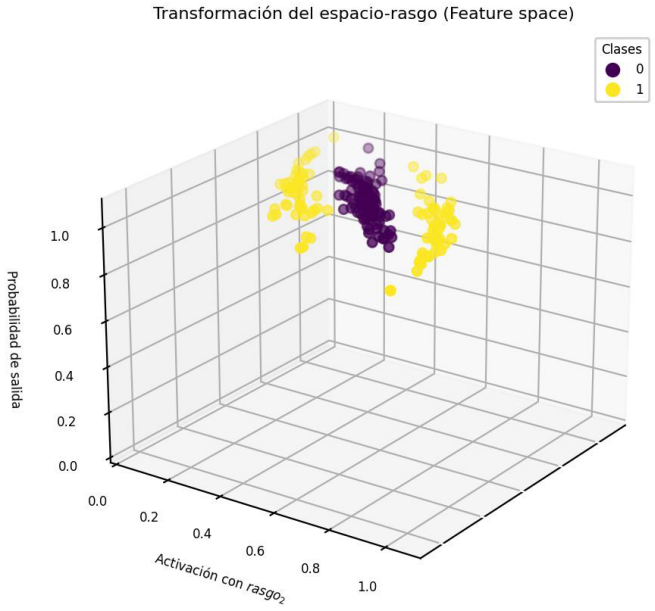


Figura C.5: Distribución de datos inicial

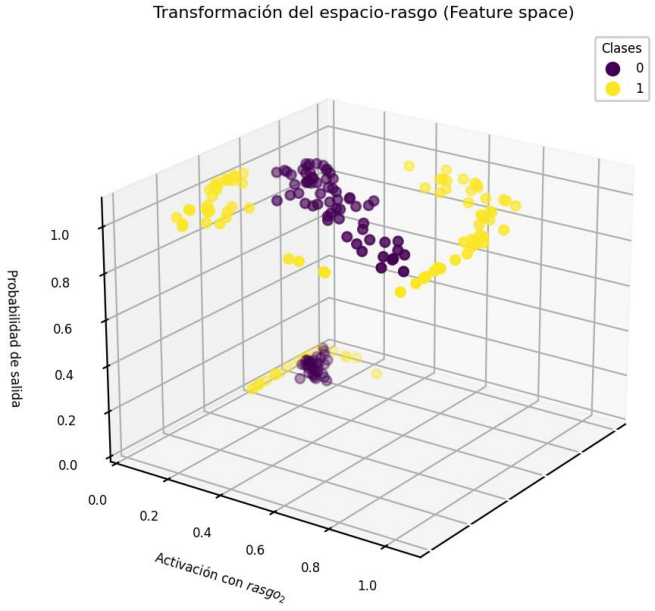


Figura C.6: Transformación inicial

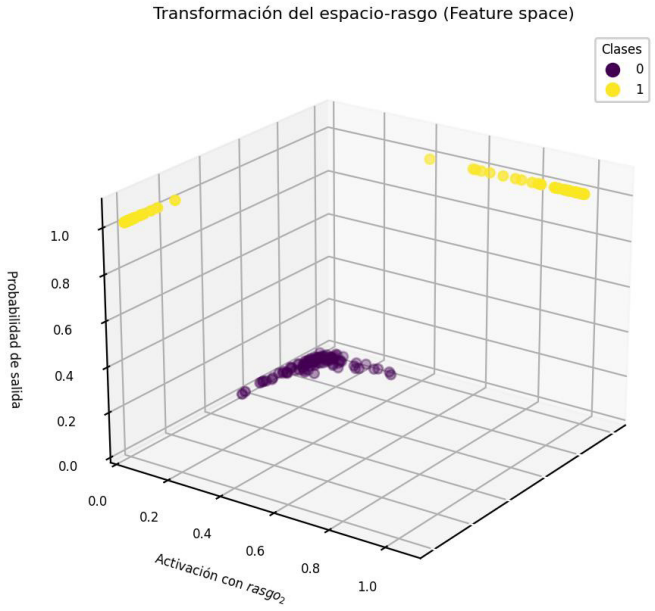


Figura C.7: Separación de nuestros datos

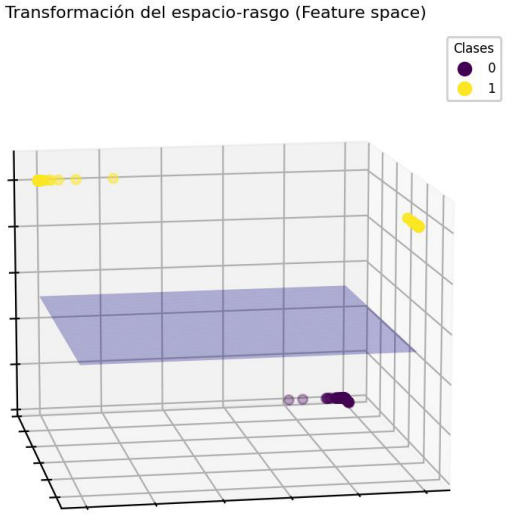


Figura C.8: Plano que separa nuestros datos (umbral de decisión)

Entonces, tenemos la distribución de datos mostrada en C.5, a través de la cual no es fácil trazar un plano. En la secuencia de C.6 hasta C.8 [5], podemos ver lo que le sucede a nuestros datos durante el entrenamiento al ser procesados por la red. Al terminar este procesamiento de la información, podemos observar que podemos separar nuestros datos muy fácilmente con un plano. Todo esto se logra a través del entrenamiento de la red, donde entran en juego la función de costo, el optimizador, los datos y los hiperparámetros.

La figura C.7 nos muestra cómo la red clasifica la información que le dimos al inicio, es decir, una categoría se clasificará con una probabilidad de 0.5-1 y la otra categoría de 0.49-0. Idealmente, en este caso, seríamos capaces de separar los datos sin mucho error. Sin embargo, en la vida real, no es tan fácil para estos sistemas separar los datos limpiamente, ya que hay varios factores, como los mencionados en el párrafo anterior, que complican el proceso.

[5]Se han removido las marcas y títulos de los ejes con propósitos estéticos.

Apéndice D

Resultados de las redes

A continuación se presentan los resultados de las redes cuando se analizó el efecto de cambiar el valor del umbral de decisión U (Se presenta como 1-U debido a que la red diseñada toma como 1 el valor de la clase NB), para elegir la mejor en base a la necesidad de registrar la mayor cantidad de barreras metálicas.

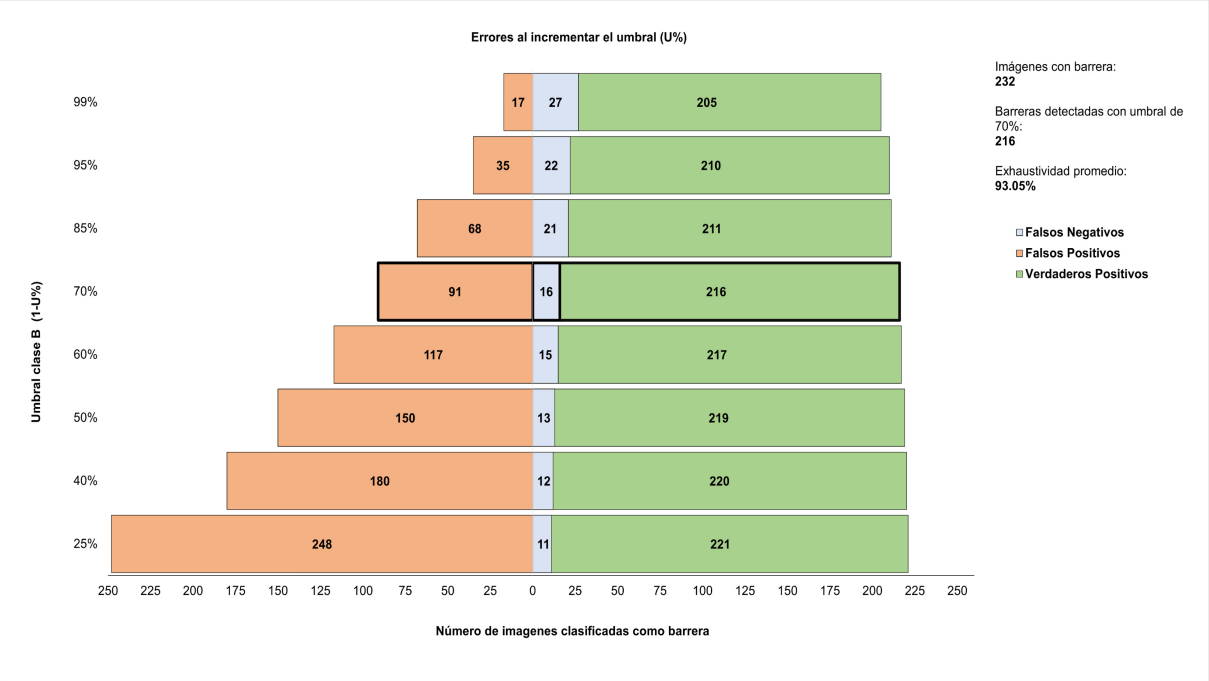


Figura D.1: Red C-mod

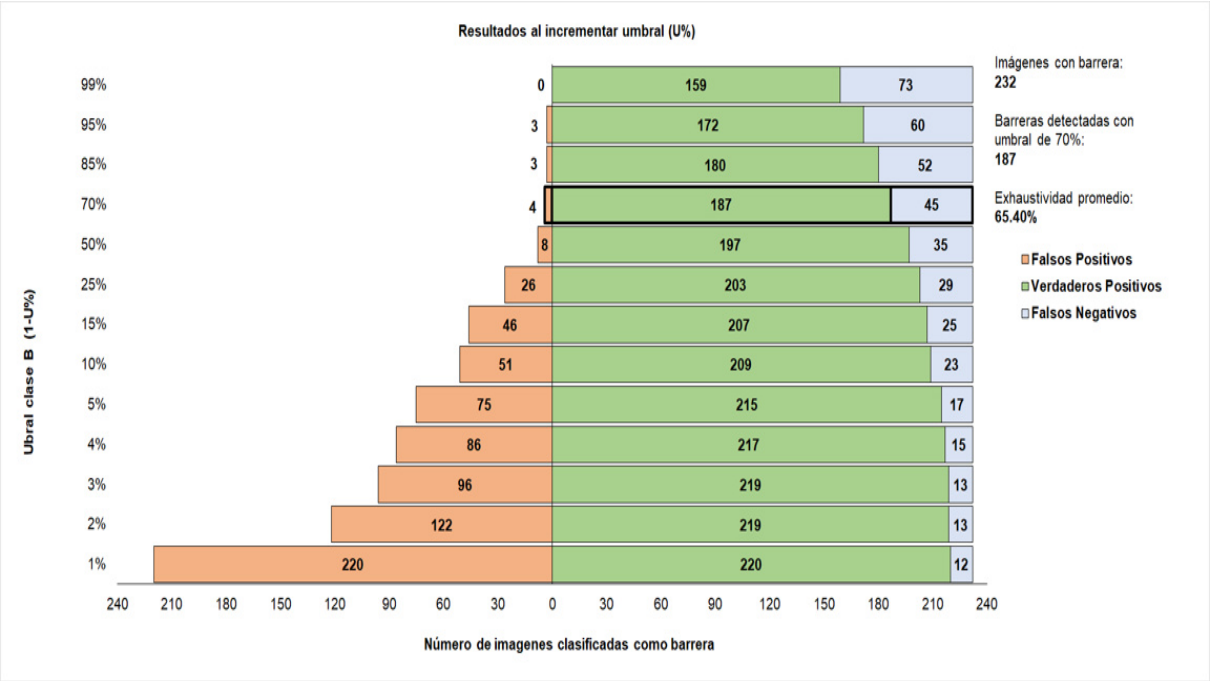


Figura D.2: Red F-mod

Apéndice E

Plataforma de recorrido virtual

La siguiente es una captura del recorrido virtual que podemos realizar en la plataforma de la SCT:
Link :http://caminero.sctcloud.com.mx/.
Encerrado en rojo tenemos algunos datos que debemos registrar en Excel (longitud, latitud y posición actual en la carretera). La flecha roja apunta al icono que presionaríamos para avanzar en la carretera.

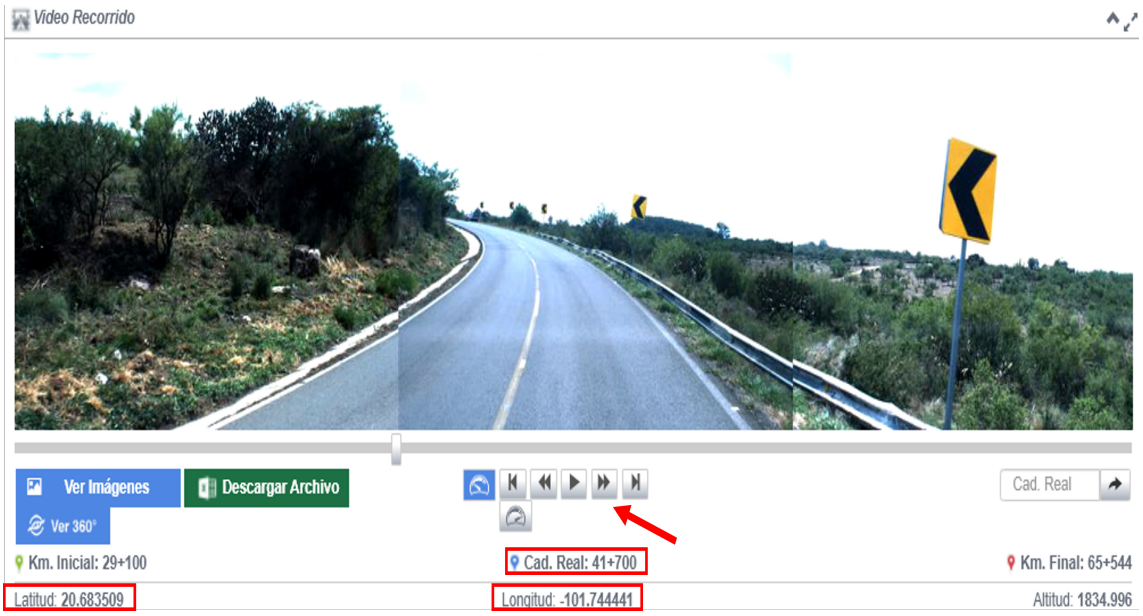


Figura E.1: Ejemplo de recorrido virtual de una carretera

Apéndice F

Formatos de salida

A continuación se presenta el formato de salida producido por el software desarrollado. Se ha seccionado en tres tablas, pero originalmente es una sola con estos 31 encabezados.

De estos 31 encabezados, 15 se pueden resolver fácilmente. Actualmente, 11 de ellos se determinan con el sistema diseñado, mientras que 4 de ellos se pueden determinar si se tiene acceso a las bases de datos. Los restantes 6 de estos 31 encabezados solamente los puede llenar quien esté haciendo el inventario, ya que son valores subjetivos. Por lo tanto, la información de 25 encabezados puede ser rellenada con un software que incorpore una red neuronal convolucional.
Los colores representan lo siguiente:

1. Rojo: Datos que conocemos desde el inicio.
2. Gris: Datos que dependen de quien esté llenando el inventario.
3. Azul: Datos que podemos determinar con una red como la diseñada.
4. Verde: Datos que se pueden encontrar con el sistema actual.
5. Amarillo: Datos que aún no pueden ser llenados con el sistema actual.

LOCALIZACIÓN DEL DISPOSITIVO DE SEGURIDAD									
No.	Carretera	Tramo	Ruta	Clasificación conforme a RPD	Cuerpo	Lado	km inicial	km final	Longitud (m)
SEC154-01									
1	IRAPUATO-TEPATITLAN	CUERÁMARO - LÍM. EDOS. GTO./JAL.	MEX-90	C	ÚNICO	ORILLA DERECHA	33920	33960	40
2	IRAPUATO-TEPATITLAN	CUERÁMARO - LÍM. EDOS. GTO./JAL.	MEX-90	C	ÚNICO	ORILLA IZQUIERDA	34380	34420	40
SEC158-02									
3	SAN LUIS DE LA PAZ-GUANAJUATO	SAN LUIS DE LAZ-DOLORES HIDALGO	MEX-110	C	ÚNICO	ORILLA DERECHA	8180	8280	100
4	SAN LUIS DE LA PAZ-GUANAJUATO	SAN LUIS DE LAZ-DOLORES HIDALGO	MEX-110	C	ÚNICO	ORILLA IZQUIERDA	8180	8280	100
SEC160-01									
5	SAN LUIS DE LA PAZ-GUANAJUATO	RAMAL A CAÑADA DE MORENO	MEX-110	C	ÚNICO	ORILLA DERECHA	11860	11860	20
6	SAN LUIS DE LA PAZ-GUANAJUATO	RAMAL A CAÑADA DE MORENO	MEX-110	C	ÚNICO	ORILLA IZQUIERDA	13860	13900	40
7	SAN LUIS DE LA PAZ-GUANAJUATO	RAMAL A CAÑADA DE MORENO	MEX-110	C	ÚNICO	ORILLA IZQUIERDA	540	540	20
8	SAN LUIS DE LA PAZ-GUANAJUATO	RAMAL A CAÑADA DE MORENO	MEX-110	C	ÚNICO	ORILLA IZQUIERDA	4320	4320	20
SEC151-01									
9	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA DERECHA	21360	21320	40
10	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA DERECHA	21260	21200	60
11	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA DERECHA	11240	11200	40
12	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA DERECHA	30560	30560	20
13	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA DERECHA	30200	30200	20
14	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA DERECHA	13000	13000	20
15	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA IZQUIERDA	22760	22740	20
16	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA IZQUIERDA	21340	21300	40
17	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA IZQUIERDA	21260	21200	60
18	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA IZQUIERDA	11240	11200	40
19	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA IZQUIERDA	35960	35960	20
20	MARAVATIO - ACAMBARO	LIM. EDOS. MICH./GTO.- ACAMBARO	MEX-61	C	ÚNICO	ORILLA IZQUIERDA	21380	21380	20

Figura F.1: Identificación de barreras

UBICACIÓN GEOGRÁFICA				TIPO DE BARRERA A EVALUAR				
Latitud inicial	Longitud inicial	Latitud final	Longitud final	Barreras	Material de la barrera	Estatus de la barrera	Observaciones	Longitud de barrera estimada que requiere sustituir o reparar en el tramo analizado
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				
				OD-4.1 BARRERA DE ORILLA DE CORONA				

Figura F.2: Tipos y ubicación

TIPO DE SECCIÓN EXTREMA QUE CUENTA LA BARRERA						TIPO DE SECCIONES DE TRANSICIÓN CON QUE CUENTA LA BARRERA DE PROTECCIÓN					
			Estado físico de la secciones extremas Al inicio / Al final						Estado Físico de la barrera de transición Al Inicio / Al final		
Al Inicio	Al final	Al Inicio	Observaciones	Al Final	Observaciones	Al Inicio	Al final	Al inicio	Observaciones	Al Final	Observaciones

Figura F.3: Extremos y transición

Bibliografía

Brownlee, J. (2019). *Difference Between a Batch and an Epoch in a Neural Network*. Recuperado el 10 de julio, 2022 de https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. Machine Learning Mastery.

colaboradores de Wikipedia (2019). *Hiperplano*. Recuperado el 13 de abril, 2022 de https://es.wikipedia.org/wiki/Hiperplano. Wikipedia, la enciclopedia libre.

colaboradores de Wikipedia (2020). *Error Cuadrático Medio*. Recuperado el 09 de mayo, 2021 de https://es.wikipedia.org/wiki/Error_cuadrÃątico_medio. Wikipedia, la enciclopedia libre.

colaboradores de Wikipedia (2020). *Homeomorfismo*. Recuperado el 13 de abril, 2022 de https://es.wikipedia.org/wiki/Homeomorfismo. Wikipedia, la enciclopedia libre.

colaboradores de Wikipedia (2021). *Propiedad topológica*. Recuperado el 13 de abril, 2022 de https://es.wikipedia.org/wiki/Propiedad_topológica. Wikipedia, la enciclopedia libre.

colaboradores de Wikipedia (2021). *Regla de la cadena*. Recuperado el 10 de mayo, 2022 de https://es.wikipedia.org/wiki/Regla_de_la_cadena. Wikipedia, la enciclopedia libre.

colaboradores de Wikipedia (2021). *Valor-F*. Recuperado el 04 de marzo, 2022 de https://es.wikipedia.org/wiki/Valor-F. Wikipedia, la enciclopedia libre.

colaboradores de Wikipedia (2022). *Desviación media*. Recuperado el 16 de mayo, 2022 de https://es.wikipedia.org/wiki/DesviaciÃşn_media. Wikipedia, la enciclopedia libre.

development team, T. M. (2022). *Choosing Colormaps in Matplotlib — Matplotlib 3.5.0 documentation*. Recuperado el 16 de mayo, 2022 de https://matplotlib.org/3.5.0/tutorials/colors/colormaps.html. Matplotlib.

Draelos, R. (2022.). Grad-cam: Visual explanations from deep networks. Recuperado el 15 de Mayo, 2022 de https://glassboxmedicine.com/2020/05/29/grad-cam-visual-explanations-from-deep-networks/. Glass Box.

Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2nd ed edition.

Ghazvineh, S., Nouri, G., Lavassani, S., Gharehbaghi, V., and Nguyen, A. (2021.). Application of 2-d convolutional neural networks for damage detection in steel frame structures. Recuperado el 26 de Diciembre, 2022 de https://doi.org/10.48550/arXiv.2110.15895. Arxiv.

Graf, S., Pagany, R., Dorner, W., and Weigold, A. (2019). Georeferencing of road infrastructure from photographs using computer vision and deep learning for road safety applications. *Proceedings of the 5th International Conference on Geographical Information Systems Theory, Applications and Management*. https://doi.org/10.5220/0007706800710076.

IBM (2021a). *Operadores lógicos*. Recuperado el 12 de abril, 2022 de https://www.ibm.com/docs/es/cognos-analytics/10.2.2?topic=SSEP7J_10.2.2/com.ibm.swg.ba.cognos.dg_rtm_wb.10.2.2.doc/c_n1602e4.html. ©Copyright IBM Corp.2014.

IBM (2021b). *Valores booleanos*. Recuperado el 12 de abril, 2022 de https://www.ibm.com/docs/es/db2woc?topic=list-boolean-values. ©Copyright IBM Corp.2014.

Innat, M. (2021). *How to implement Grad-CAM on a trained network* . Recuperado el 23 de septiembre, 2021 de https://stackoverflow.com/questions/66182884/how-to-implement-grad-cam-on-a-trained-network. Stack Overflow.

Jain, V., Gupta, P., Chaudhry, A., Batra, M., and Hemanth, D. (2022.). A modified deep convolution siamese network for writer-independent signature verification. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, page 479–498. Recuperado el 26 de Diciembre, 2022 de https://doi.org/10.1142/s0218488522400177.

Jeffrey, O., Alexander, S.,, Amir, R.,, Leonidas, J.,, and Jitendra, M. (2020.). Side-tuning: Network adaptation via additive side networks. *CoRR*, abs/1912.13503(1). https://doi.org/10.48550/arXiv.1912.13503.

Keras (2020). *Grad-CAM class activation visualization*. Recuperado el 23 de Abril, 2022 de https://keras.io/examples/vision/grad_cam/.

Keras (2022.). *Keras Applications*. Recuperado el 23 de Abril, 2022 de https://keras.io/api/layers/regularization_layers/dropout/.

Khan Academy (2022.). *Descenso de gradiente*. Recuperado el 13 de abril, 2022 de https://es.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent. https://es.khanacademy.org/.

Khandelwal, R. (2020). *Convolutional Neural Network: Feature Map and Filter Visualization*. Recuperado el 24 de junio, 2021 de https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c. Medium.

Kingma, D.P. and Ba, J. (2014.). Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editor, *3rd International Conference on Learning Representations, May 7-9, 2015, Conference Track Proceedings*, ICLR 2015, page 1, San Diego, CA, USA. arXiv. https://doi.org/10.48550/arXiv.1412.6980.

Labach, A., Salehinejad, H., and Valaee, S. (2019.). Survey of dropout methods for deep neural networks. Recuperado el 23 de Marzo, 2022 de https://www.researchgate.net/publication/332774314_Survey_of_Dropout_Methods_for_Deep_Neural_Networks.

Landau, L. J. (1998). *Concepts for Neural Networks: A Survey.*, chapter 2, pages 6–8. Springer. Recuperado el 01 de abril, 2022 de https://nms.kcl.ac.uk/ton.coolen/published/1998/summerschool98.pdf.

Marcelino, P. (2018.). Transfer learning from pre-trained models-towards data science. Recuperado el 16 de Mayo, 2022 de https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751. Medium.

Mitiku. (2014). How to do transfer-learning on our own models?. Recuperado el 16 de Mayo, 2022 de https://stackoverflow.com/questions/51336761/how-to-do-transfer-learning-on-our-own-models. Stack Overflow.

Mohamad, A., Harris, D., and Miller, G. (2019.). Robust pixel-level crack detection using deep fully convolutional neural networks. Recuperado el 26 de Diciembre, 2022 de https://doi.org/10.1061/(ASCE)CP.1943-5487.0000854. Journal of Computing in Civil Engineering.

Olah, C. (2014a). *Calculus on Computational Graphs: Backpropagation – colah's blog.* Recuperado el 31 de marzo, 2022 de http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/. Colah.Github.Io.

Olah, C. (2014b). *Neural Networks, Manifolds, and Topology – colah's blog.* Recuperado el 31 de marzo, 2022 de http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/. Colah.Github.Io.

Phan, B. (2021). *10 Minutes Building a CNN Binary Image Classifier in TensorFlow.* Recuperado el 17 de junio, 2021 de https://towardsdatascience.com/10-minutes-to-building-a-cnn-binary-image-classifier-in-tensorflow-4e216b2034aa. Medium.

Purva (2020). *Precision vs. Recall – An Intuitive Guide for Every Machine Learning Person.* Recuperado el 10 de Mayo, 2022 de https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/. Analitics Vidhya.

Ramprasaath, S., Michael, C., Abhishek, D., Ramakrishna, V., Devi, P., and Dhruv, B. (2019.). Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359. https://doi.org/10.48550/arXiv.1610.02391.

Rezapour, M. and Ksaibati, K. (2021.). Convolutional neural network for roadside barriers detection: Transfer learning versus non-transfer learning. *Signals*, page 72–86. https://doi.org/10.3390/signals2010007.

Sainju, A. M. and Jiang, Z. (2020.). Mapping road safety features from streetview imagery. *ACM/IMS Transactions on Data Science*, pages 1–20. https://doi.org/10.1145/3362069.

Sarhan, A. M. (2020.). A modified deep convolution siamese network for writer-independent signature verification. *Journal of Advances in Medicine and Medical Research*, page 15–26. Recuperado el 26 de Diciembre, 2022 de https://doi.org/10.9734/jammr/2020/v32i1230539.

Sen, J. and Mehtab, S. (2020.). Stock price prediction using convolutional neural networks on a multivariate timeseries. Recuperado el 26 de Diciembre, 2022 de https://doi.org/10.36227/techrxiv.15088734.v1. National Conference on Machine Learning and Artificial Intelligence.

Team, K. (2022). Keras documentation: Spatialdropout2d layer. Recuperado el 15 de Mayo, 2022 de https://keras.io/api/layers/regularization_layers/spatial_dropout2d/. Keras.io.

Tegegne, A. M. (2022.). Applications of convolutional neural network for classification of land cover and groundwater potentiality zones. *Journal of Engineering*, pages 1–8. Recuperado el 26 de Diciembre, 2022 de https://doi.org/10.1155/2022/6372089.

The BBC (2022). *Calculating percentage increase and decrease*. Recuperado el 10 de julio, 2022 de https://www.bbc.co.uk/bitesize/guides/zpjmjty/revision/2.

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015.). Efficient object localization using convolutional networks. Recuperado el 15 de Mayo, 2022 de https://arxiv.org/pdf/1411.4280.pdf. arXiv.

Wikipedia contributors (2022). *Transfer learning*. Recuperado el 04 de Marzo, 2022 de https://en.wikipedia.org/wiki/Transfer_learning. Wikipedia.